UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Mejora de voz mediante redes neuronales profundas

Carlos Lamas Álvarez

Tutor: Doroteo Torre Toledano

Junio 2017

Mejora de voz mediante redes neuronales profundas

AUTOR: Carlos Lamas Álvarez

TUTOR: Doroteo Torre Toledano

AUDIAS – AUdio, Data Intelligence And Speech

Dpto. Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Junio de 2017

Resumen (castellano)

En este Trabajo Fin de Grado se ha implementado un sistema de limpieza de ruido, en señales de voz, basado en redes neuronales profundas (DNNs). Con esto en mente se consiguió generar un sistema parametrizable y aplicable a diferentes objetivos, más allá del mencionado en este TFG.

Como inspiración en el desarrollo y diseño de este proyecto se ha partido de un trabajo publicado en 2015 en *IEEE Transactions on Audio, Speech, and Language Processing*, titulado '*A Regression Approach to Speech Enhancement Based on Deep Neural Networks*' y escrito por Chin-Hui Lee, Young Xu, Jun Du y Li-Rong Dai.

Para la realización de este trabajo se ha requerido de la implementación de código en Python, más concretamente el uso de las librerías Theano y Blocks. Theano constituye una de las herramientas más extendidas en la implementación de sistemas de *machine learning* pero, dada su complejidad, se decidió compaginarla con la otra librería citada, Blocks. Esta actúa por encima de Theano y hace más sencilla la interpretación y manipulación de las herramientas.

Se han usado dos bases de datos principales, para de señales de voz se ha utilizado TIMIT y para señales de ruido se ha utilizado HU; ambas tanto para entrenamiento como para pruebas, aunque se harán ciertos experimentos con otras bases de datos como son ALBAYZIN, para voz, y NOISEX, para ruido. Por otro lado, las bases de ruidos solo se usarán en el 'ensuciado' de los audios empleados para entrenar y para experimentos. Este 'ensuciado' se ha conseguido implementar mediante el uso de la herramienta FANT desarrollada en el proyecto AURORA, un software que permite la adición de señales de ruido a señales de voz con un *Signal to Noise Ratio* (SNR) controlable.

Para la evaluación del sistema se ha utilizado el estándar de estimación objetiva de la calidad subjetiva PESQ de la ITU-T. El objetivo al usar esta herramienta es el de mejorar los resultados devueltos por ella para audios extraídos de la red neuronal con respecto a los audios ruidosos originales.

Abstract (English)

This Bachelor Thesis has implemented a noise cleaning system, on voice signals, based on deep neural networks (DNNs). With this in mind, it has been achieved the generation of a parametrizable system that can be focused on different tasks, not only on the one that has been proved in this project.

It has been taken, as inspiration for this project, the paper published in 2015 in *IEEE Transactions on Audio, Speech, and Language Processing*, titleled '*A Regression Approach to Speech Enhancement Based on Deep Neural Networks*' and written by Chin-Hui Lee, Young Xu, Jun Du y Li-Rong Dai.

Python has been used on the realization of this work; to be more precise, the libraries Thenao and Blocks. Thenao is one of the best and more used, around the world, tools in *machine learning* problems. Because of its complexity, it has been decided to use Blocks as well. This library works on a higher level than Theano and helps in the manipulation of the tools.

There are two databases that have been used as main, TIMIT as voice database and HU as noise database, either train and test; however, other databases like ALBAYZIN, in voice, and NOISEX, in noise, have been used as well in test trials. Noise databases have been only used in the noise addition process for train and test. This process has been possible thanks to the FANT tool by AURORA, a software that allows users to add noise to a voice signal with a particular *Signal to Noise Ratio* (SNR).

For the evaluation process, it has been used the standard of objective estimation of the subjective quality PESQ, by the ITU-T. The objective using this tool is to enhance the results thrown by it on the audio signals, extracted from the DNN, compared with the same result based on the original noisy voice signals.

Palabras clave (castellano)

Mejora de voz, red neuronal profunda, calidad de voz, espectro, PESQ.

Keywords (inglés)

Speech enhancement, deep neural network, voice quality, spectrum, PESQ.

Agradecimientos

En primer lugar, querría dar las gracias a mi tutor Doroteo Torre Toledano por darme la oportunidad de realizar este proyecto y estar siempre disponible para cualquier inquietud que pudiera tener durante la realización del mismo. También me gustaría agradecerle, de una manera muy especial, a Juan Maroñas Molano, pues es él quien me enseñó todo lo necesario y tuvo la paciencia de aguantar a un estudiante tan perdido como yo durante el curso; sin él este TFG nunca habría acabado como lo ha hecho, de verdad, muchas gracias.

Por otro lado, quería agradecer a todas las personas del laboratorio ATVS por la cercanía, la paciencia y la ayuda que me han prestado durante todo el curso, en especial agradecérselo a Adrián y a Rubén Zazo porque desde que me dejaron hacerles una pregunta no he parado ni un solo día. También agradecérselo a mis amigos y a mi 'Concilio de los Telecos' porque consiguieron serenarme cuando más agobiado estaba e hicieron que no perdiera el norte, muy especialmente darle las gracias a Álvaro Iglesias Arias pues en estos dos últimos años ha demostrado ser un apoyo constante e imprescindible, nunca olvidaré aquel *append*…

Por último, un gracias enorme a mi familia**,** por estar siempre dispuestos a ayudarme y mostrar siempre la confianza y el apoyo que necesitaba para no tirar todo por la borda, os quiero.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN 1

1.1 MOTIVACIÓN 1
1.2 OBJETIVOS 2
1.3 ORGANIZACIÓN DE LA MEMORIA 2

2. ESTADO DEL ARTE 3

2.1 ESTADO DEL ARTE EN TÉCNICAS DE MEJORA DE VOZ 3
2.1.1 Reducción de ruido por filtrado 3
2.1.2 Reducción de ruido por restauración espectral 4
2.1.3 Reducción de ruido por modelos 4
2.2 ESTADO DEL ARTE EN REDES NEURONALES 6
2.2.1 Funciones de activación 8
2.2.2 Función de coste y backpropagation 10
2.2.3 Matrices de pesos, vectores bias, tasa de aprendizaje y momentum 11
2.2.4 Batches y epochs 12

3. DISEÑO Y DESARROLLO DEL TFG 13

3.1 PREPARACIÓN PREVIA AL ENTRENAMIENTO DE LA SEÑAL DE VOZ 13
3.1.1 Enventanado y cálculo del espectro 14
3.1.2 Acondicionamiento y guardado de los datos 15
3.2 DESARROLLO DEL GRAFO Y DEFINICIÓN DE LOS PARÁMETROS DE LA RED NEURONAL 15
3.2.1 Elección de tamaño de grafo 16
3.2.2 Elección de vectores de bias y matrices de pesos 16
3.2.3 Funciones de activación y función de coste 18
3.2.4 Elección del tamaño de batch y número de epochs 18
3.2.5 Implementación y entrenamiento la red neuronal 19
3.3 RECONSTRUCCIÓN DE LA SEÑAL LIMPIA 20
3.4 PESQ, TÉCNICA DE VALORACIÓN 20

4. PRUEBAS Y RESULTADOS 23

4.1 RESULTADOS PESQ PARA DIFERENTES TAMAÑOS DE BASE DE DATOS Y 50 EPOCHS ... 23
4.2 RESULTADOS PESQ PARA DIFERENTES EPOCHS Y 90H DE BASE DE DATOS 27
4.3 RESULTADOS PESQ PARA RUIDOS NOISEX 29
4.4 RESULTADOS PESQ PARA BASE DE DATOS DE VOZ ALBAYZIN SIN CAMBIO DE RUIDOS 31

5. CONCLUSIONES Y TRABAJO FUTURO 33

6. REFERENCIAS 35

7. GLOSARIO 37

ÍNDICE DE FIGURAS

FIGURA 1: MODELO DE PREDICCIÓN LINEAL 5

FIGURA 2: EJEMPLO DE TOPOLOGÍA HMM: BAKIS (IZQUIERDA) Y ERGÓDICA (DERECHA) 6

FIGURA 3: EJEMPLO DE TOPOLOGÍA NO DETALLADA DE UNA RED NEURONAL. EN ESTE EJEMPLO EL VECTOR DE ENTRADA $\boldsymbol{x} \in \mathbb{R}\boldsymbol{n}$, EL VECTOR DE SALIDA $\boldsymbol{y} \in \mathbb{R}\boldsymbol{u}$, Y AL MENOS 2 CAPAS INTERMEDIAS CON SUS RESPECTIVAS DIMENSIONES R Y C 7

FIGURA 4: REPRESENTACIÓN DE DIFERENTES FUNCIONES DE ACTIVACIÓN PARA UN INTERVALO DE VALORES DE UN VECTOR ENTRANTE [-10, 10] 9

FIGURA 5: ESQUEMA DE LAS FASES MÁS RELEVANTES DEL PROYECTO. 13

FIGURA 6: COMPARATIVA DE VENTANAS HAMMING PARA DIFERENTES GANANCIAS C 14

FIGURA 7: EJEMPLO DE ESPECTROGRAMA DE UN AUDIO LIMPIO 16

FIGURA 8: ILUSTRACIÓN DEL GRAFO IMPLEMENTADO EN EL TFG 17

FIGURA 9: MODELO DE PERCEPCIÓN UTILIZADO POR PESQ 21

FIGURA 10: EJEMPLOS DE ESPECTROGRAMAS SUCIOS (IZQUIERDA) Y LIMPIADOS CON UNA RED DE 90H DE BASE DE DATOS Y 50 EPOCHS (DERECHA) PARA UN MISMO AUDIO ORIGINAL (ARRIBA) 25

FIGURA 11: EJEMPLO DE LA DISTRIBUCIÓN GAUSSIANA DE VALORES PESQ PARA UN CASO DE SNR = 5 26

FIGURA 12: EJEMPLO DEL MALFUNCIONAMIENTO EN DISTRIBUCIÓN DEL PESQ EN EL SISTEMA PARA SEÑALES DE SNR = 10 26

FIGURA 13: EJEMPLOS ESPECTROGRAMAS: AUDIO ORIGINAL (ARRIBA), AUDIO RUIDOSO ORIGINAL (DEBAJO DEL ANTERIOR) Y AUDIOS LIMPIADOS CON 90H Y DIFERENTES EPOCHS. 28

FIGURA 14: EJEMPLO DE ESPECTROGRAMAS CON BASE DE DATOS DE RUIDOS NOISEX: ORIGINAL (ARRIBA), RUIDOSO ORIGINAL (MEDIO) Y PROCESADO POR LA RED NEURONAL (ABAJO). 30

FIGURA 15: EJEMPLO DE ESPECTROGRAMAS CON BASE DE DATOS DE VOZ ALBAYZIN: ORIGINAL (ARRIBA), RUIDOSO ORIGINAL (MEDIO) Y PROCESADO POR LA RED NEURONAL (ABAJO). 32

ÍNDICE DE TABLAS

TABLA 1: RESULTADOS DE MEDIA DE PESQ OBTENIDOS PARA LA BASE DE DATOS DE TEST DE AUDIOS RUIDOSOS SIN MEJORAR..... 23

TABLA 2: RESULTADOS DE MEDIA DE PESQ OBTENIDOS PARA DIFERENTES TAMAÑOS DE BASES DE DATOS Y MISMO NÚMERO DE EPOCHS DE ENTRENAMIENTO..... 24

TABLA 3 RESULTADOS DE MEDIA DE PESQ PARA UNA BASE DE DATOS DE 90H Y UNA VARIACIÓN DE LAS EPOCHS DE ENTRENAMIENTO..... 27

TABLA 4: RESULTADOS DE MEDIA DEL PESQ PARA AUDIOS RUIDOSOS CON NOISEX, PARA AUDIOS LIMPIADOS CON RED DE 90H Y 50 O 100 EPOCHS..... 29

TABLA 5: RESULTADOS DE MEDIA DE PESQ PARA AUDIOS RUIDOSOS DE ALBAYZIN Y AUDIOS LIMPIADOS CON 90H Y 50 EPOCHS..... 31

1. Introducción

1.1 Motivación

Hoy en día es notoria la gran dependencia tecnológica que sufre el ser humano. Cada vez son más los dispositivos con lo que se interactúa y, a la larga, estos pueden provocar conflictos si no están perfectamente optimizados.

Uno de estos conflictos podría ser la influencia del ruido en las señales de audio. Como se ha mencionado antes, cada vez hay más dispositivos y muchos de ellos son capaces de realizar grabaciones de audio. Es de suponer que no todos ellos van a estar dotados de los mejores mecanismos de captación y por ende los resultados de estos no serán de gran calidad.

Podría parecer que por sí mismo no sea algo muy relevante; sin embargo, si se considera que estos audios pueden ser usados en tareas como: reconocimiento de locutor, reconocimiento de habla o incluso aplicaciones forenses, el ruido podría suponer un problema muy a tener en cuenta.

En la actualidad existen muchas técnicas viables para la limpieza de ruido en señales de audio, algunas de las cuales serán expuestas más adelante. Sin embargo, en los últimos años se abrió un nuevo campo de interés para problemas de esta índole con la evolución de las técnicas de *machine learning*, o aprendizaje de máquina, llamadas DNN ('*Deep Neural Networks*' o Redes Neuronales Profundas).

Las redes neuronales permiten estructurar modelos computacionales compuestos por múltiples capas de 'neuronas', siendo estas capaces de aprender representación de datos con muchos más niveles de abstracción de los que se consiguen con técnicas más tradicionales. Este tipo de modelos ha mejorado su rendimiento drásticamente en los últimos tiempos, en especial en problemas de reconocimiento tanto de habla como de imagen, aunque en esto último no entraremos en este TFG. [1]

El aprendizaje profundo nos descubre estructuras en grandes bases de datos mediante el uso de la técnica denominada '*backpropagation*'. Esta técnica, que a efectos prácticos es un descenso por gradiente, permite a una red definir como deberían ser los diferentes parámetros de la estructura para poder hacer una representación óptima de datos en cada capa, estando estas influenciadas con las anteriores.

Por ello, será muy interesante ver como se comportarán dichas redes, dado su carácter no lineal, si se le pide que, en vez de problemas de clasificación típicos, procesen señales de audio con el fin de eliminar el ruido que pudiera haber en ellas. Para ello se requerirán grandes cantidades de datos de entrenamiento a fin de que aporten una gran variabilidad de situaciones durante el entrenamiento y así se amplíe su rango de acción.

Cabe destacar que para la realización de este proyecto se ha tenido muy en cuenta el trabajo titulado '*A Regression Approach to Speech Enhancement Based on Deep Neural Networks*', que ofrece una perspectiva similar a la que se busca en este trabajo. [2]

1.2 Objetivos

En este proyecto se pretende demostrar que las redes neuronales profundas ofrecen grandes resultados en la limpieza de ruido en señales de voz. Pese a que en algunos casos los tiempos de entrenamiento son muy extensos, este hecho carece de importancia al necesitar entrenar tan solo una vez. Por ello se tratará de implementar una red neuronal profunda viable y no muy compleja, para facilitar su uso en futuros trabajos.

1.3 Organización de la memoria

Este trabajo se ha estructurado por capítulos, tratando de ofrecer de la manera más clara posible toda la información necesaria para su entendimiento. Así pues, la memoria queda estructurada de la siguiente manera.

- Introducción: como acaban de leer, en este capítulo se han expuesto la motivación y los objetivos más remarcables del proyecto.

- Estado del arte: en este capítulo se muestra, de manera breve, al lector las diferentes técnicas que se han usado, o se usan, en la actualidad para la limpieza de ruido en señales de voz. Seguidamente se hará un desarrollo más extenso del funcionamiento de las redes neuronales, así como sus características principales y la razón de su uso.

- Diseño y desarrollo del trabajo: este bloque expondrá de manera extensa todos los pasos que han tenido que llevarse a cabo para poder diseñar e implementar la red neuronal. También se hará hincapié en las técnicas que se han usado para poder ejecutar y entrenar de manera adecuada dicha red.

- Pruebas y resultados: esta sección sacará a la luz los resultados obtenidos en función de parámetros como el tiempo de entrenamiento o el tamaño de la base de datos de entrenamiento.

- Conclusiones y trabajo a futuro: se enuncian las conclusiones y se presenta el posible trabajo a futuro.

2. Estado del Arte

2.1 Estado del arte en técnicas de mejora de voz

En este capítulo se hará un repaso por las técnicas de limpieza de voz previas a la aparición de las redes neuronales. Para ello se explicarán de manera breve las más relevantes en la industria durante los últimos años.

2.1.1 Reducción de ruido por filtrado

Esta técnica pretende pasar una señal ruidosa por un filtro lineal. Estos filtros son parametrizables, una cualidad que permite ajustarlos lo más posible y poder atenuar el ruido sin introducir mucha distorsión. Típicamente se utilizan dos modelos de filtros diferentes para esta tarea: [3]

- **Filtro de Wiener en tiempo**: cosiste la técnica más empleada. El filtro puede obtenerse minimizando el MSE (*Mean Square Error* o Error Cuadrático Medio) existente entre la señal de voz sin ruido y la señal de voz estimada de la señal ruidosa. Esta señal estimada puede calcularse aplicando un filtro FIR lineal a la señal de voz ruidosa:

$$\hat{x}(n) = h^T * y(n) \quad \text{siendo} \quad h = [h_0\ h_1\ h_2\ \dots\ h_{L-1}]$$

Con esto uno puede calcular el filtro de Wiener en tiempo mediante el cálculo de la siguiente expresión:

$$h_o = argmin\big(J_x(h)\big) \quad \text{siendo} \quad J_x(h) = MSE\{\big(x(n) - \hat{x}(n)\big)\}$$

A partir de aquí se deberá minimizar ese MSE, pero habrá un problema y es que para ello se deben utilizar las ecuaciones de Wiener-Hopf y estas hacen uso de la correlación cruzada entre la señal ruidosa y la limpia, y la limpia no es conocida. Pese a todo, este filtro presenta otro inconveniente; si se aumenta la capacidad de eliminación de ruido, aumentando el SNR (*Signal to Noise Ratio*), se introduce una distorsión a la señal resultante. Cabe destacar que se ha conseguido un modelo del filtro que consigue controlar el compromiso entre eliminación de ruido y distorsión, el filtro de Wiener Subóptimo.

- **Filtro de Wiener en frecuencia**: es un filtro basado en la densidad espectral en el dominio de la frecuencia de la señal de voz $P(\omega k)$. Sin embargo, este filtro depende de la densidad espectral de la señal limpia, este dato no es uno al que se tenga acceso en la mayoría de las ocasiones y por ello, al igual que sucede en el filtro de Wiener en tiempo, se deberá asumir que la señal de voz y el ruido están incorrelados, por lo tanto, el filtro en frecuencia podrá aproximarse de la siguiente manera:

$$H_o(\omega_k) = \frac{P_y(\omega_k) - P_v(\omega_k)}{P_y(\omega_k)}$$

$$y => \text{Señal ruidosa}, \qquad v => \text{Señal solo ruido}$$

  Este filtro tiene una versión paramétrica que permite modificar hasta tres parámetros a fin de conseguir las prestaciones requeridas para el filtro.

Cabe destacar que, para ambos tipos de filtros, tanto tiempo como frecuencia, siempre es necesario estimar el espectro del ruido en base a las zonas de la señal de voz ruidosa que solo presenten ruido. Estas zonas deben buscarse mediante el uso de un detector de actividad de voz.

2.1.2 Reducción de ruido por restauración espectral

Esta técnica consiste en suponer una buena estimación del espectro que debería tener la señal de voz limpia a partir del espectro que se tiene con la señal de voz ruidosa. Para poder llevar a cabo este proceso se usan teorías de estimación y también se asumen distribuciones estadísticas para la propia señal de voz y el ruido.
Por lo general se aplicará como estimador el MMSE (*Minimun Mean Square Error* o Mínimo Error Cuadrático Medio) y se usarán distribuciones Gaussianas. Sin embargo, existen otras opciones como estimadores ML o MAP aplicados a otras distribuciones no Gaussianas.
Al igual que se vio con los filtros de Wiener, en este problema también es común el estimar el espectro del ruido a partir de la señal ruidosa.

2.1.3 Reducción de ruido por modelos

Dentro de esta categoría se pueden destacar muchos, tal vez los que más destacan sean: los modelos armónicos, los modelos de predicción lineal y las HMMs. Es importante mencionar que todos ellos describen la estructura de la voz, algo muy importante pues se

puede usar estos conocimientos para facilitar la separación de la señal de voz y el ruido. El funcionamiento, de los modelos antes mencionados, es bastante distinto entre unos y otros: [4]

- **Modelo armónico**: modela la voz como una suma de sinusoides en frecuencias múltiplos de la frecuencia fundamental de la voz en cada instante temporal, a estas frecuencias se las conoce como armónicos. Al asumir esta aproximación de la voz, se utiliza un filtro de peine que, en teoría, solo permitirá pasar las frecuencias armónicas con un mínimo impacto del ruido. Sin embargo, requiere una estimación muy precisa de la frecuencia fundamental y por lo tanto no es un modelo válido para sonidos no sonoros; es por ello que no tiene en general muy buenos resultados.

- **Modelo de predicción lineal**: este modelo se basa en asumir que, durante los momentos en los que la voz puede considerarse pseudo-estacionaria, esta está generada por el modelo de producción lineal representado en la Figura 1.
  La parte más importante de este sistema es el modelo de tracto vocal $V(z)$, este se puede modelar como una secuencia de tubos acústicos de diferente área estimables con los coeficientes PARCOR calculables gracias al método Levinson-Durbin y a los coeficientes el método de producción lineal de voz o LPC. Finalmente se puede extraer el modelo mediante la siguiente formulación:

$$V(z) = \frac{G}{1 - \sum_{k-1}^{N} a_k z^{-1}}$$

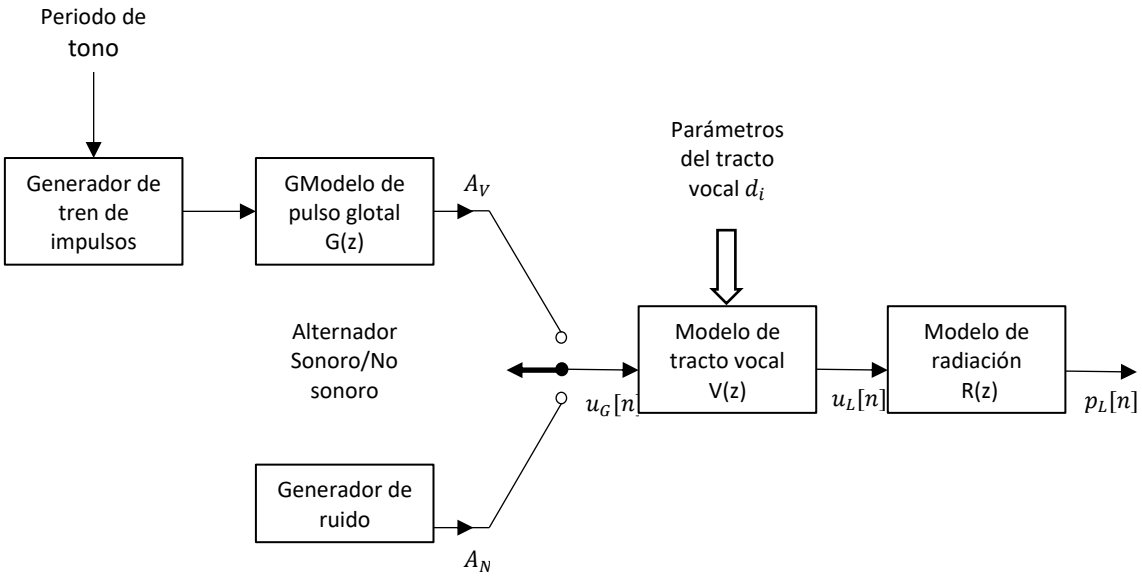

Figura 1: Modelo de predicción lineal

- **Modelo oculto de Markov o HMM**: para limpieza de ruido se deben usar dos modelos diferentes, uno para voz y otro para ruido. Ambos modelos deben entrenarse en condiciones lo más parecidas posible la situación real. Cabe destacar que constituyen una técnica muy compleja computacionalmente lo que no facilita su implementación. A su vez constan de diferentes topologías según la aplicación, algunas de las más representativas son la topología Bakis y la ergódico, Figura 2.

Algo que une a todas las técnicas vistas hasta el momento es que el ruido a considerar se supone casi siempre blanco y, por ende, de potencia constante para todas las frecuencias. Esto es importante pues, si por algún casual se conocen las características espectrales del ruido que afecta a la señal de voz, se podrán utilizar técnicas más simples como un simple filtrado o un sistema de separación de fuentes para eliminar a este de la señal de voz.

2.2 Estado del arte en redes neuronales

Las redes neuronales constituyen la técnica más extendida en resolución de problemas relacionados con el reconocimiento y el *machine learning*. Como su nombre indica, están inspiradas en el funcionamiento de las neuronas dentro del propio cuerpo humano y su capacidad para reconocer patrones. Para lidiar con esto, la información debe ser procesada y representada de una manera apropiada. [5]

Se pueden definir las redes neuronales como proyecciones algebraicas de vectores de entrada x, de dimensión parametrizable, a vectores de salida y, también parametrizables. El vector de salida y se podría definir de manera matemática de una manera sencilla: [6]

$$y = f(x) \quad donde \quad x \in \mathbb{R}^k, y \in \mathbb{R}^{k'}; \quad k, k' \geq 1$$

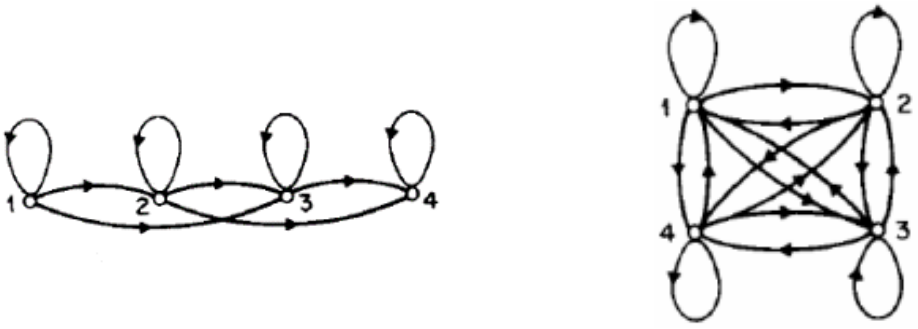

Figura 2: Ejemplo de topología HMM: bakis (izquierda) y ergódica (derecha)

En las redes neuronales existen diferentes topologías por lo general elegibles en función de las necesidades del diseñador. Cada capa oculta representada en la Figura 3 se corresponde con un vector intermedio de dimensión parametrizable al igual que los vistos anteriormente.

Por lo general en las redes neuronales, las operaciones que se llevan a cabo durante el entrenamiento son una mezcla de cálculos lineales y no lineales. El resultado calculado en cada neurona, estructura básica de una red neuronal referenciada con un círculo en la Figura 3, se lleva a cabo mediante una operación lineal por cada una de las neuronas de la capa anterior que contribuyen a esta y una posterior operación, por lo general no-lineal, aplicada al resultado llamada función de activación, $G(\cdot)$, de la cual se hablará más adelante. Con este contexto, se pueden entender las operaciones llevadas a cabo de la siguiente manera:

$$h_x^i = G(w^n * X + b^n) \quad donde \quad X \in \mathbb{R}^k; h_x^i, b^n \in \mathbb{R}^{k'};\ w^n \in \mathbb{R}^{k*k'}$$

Donde w^n se refiere a la matriz de pesos y b^n al vector bias, de ambos se hablará más adelante. X no se refiere necesariamente al vector de entrada a la red neuronal, podría ser

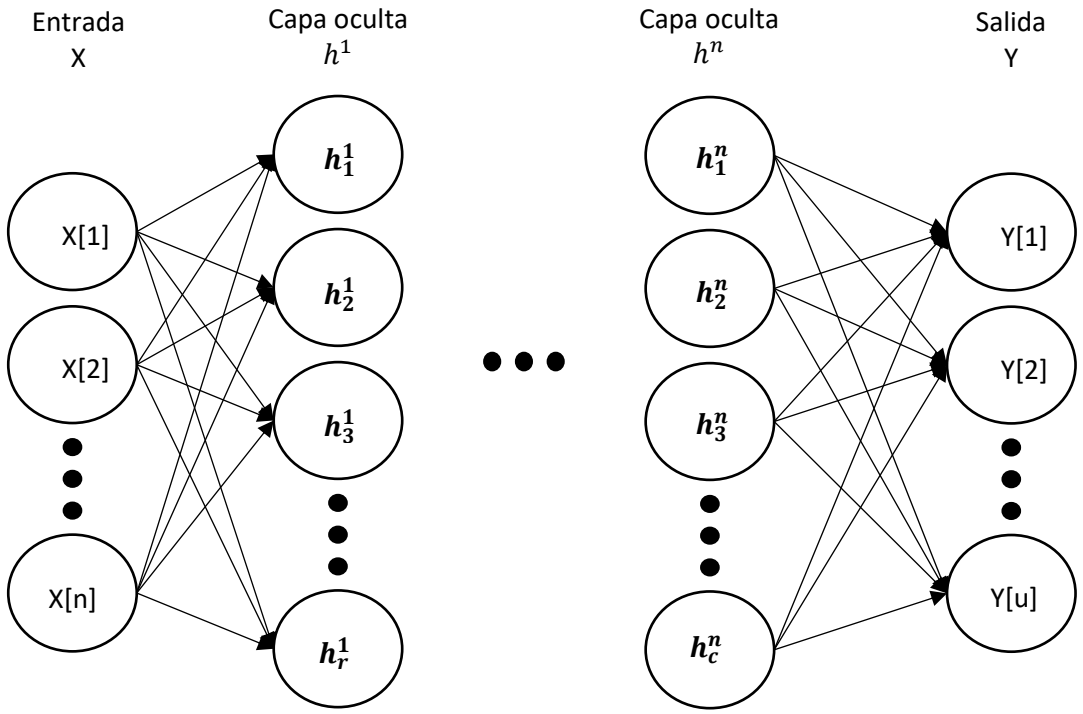

Figura 3: Ejemplo de topología no detallada de una red neuronal. En este ejemplo el vector de entrada $x \in \mathbb{R}^n$, el vector de salida $y \in \mathbb{R}^u$, y al menos 2 capas intermedias con sus respectivas dimensiones r y c

cualquier vector de salida de cualquiera de las capas que constituyen la red, excepto el vector de salida.

2.2.1 Funciones de activación

Las funciones de activación son elementos, por lo general no-lineales, que se operan al final de cada proceso en cada neurona, para definir de manera adecuada la salida de la misma. Existen una gran variedad y se eligen en función de las necesidades del diseñador. Algunas de las funciones más extendidas son: [5]

- **Sigmoide:** función delimitante no-lineal. Devuelve un valor entre 0 y 1, ver en la Figura 4, dependiendo del valor de entrada del vector al que se le aplica. Suele ser muy útil, como función de activación de salida, en problemas de clasificación binaria dadas las limitaciones de sus resultados. Es fácilmente calculable con la siguiente formulación matemática:

$$G(x_i) = \frac{1}{1 + e^{-x_i}} \quad donde \quad x_i \in \mathbb{R}^k$$

- **ReLU (Rectifier Liner Unit):** función no-lineal que permite a una red obtener de manera sencilla pocas representaciones e impedir que los pesos de una capa, w_n, se queden estancados durante el entrenamiento. Típicamente se utiliza en problemas con topologías profundas. Consiste en escoger el máximo valor por muestra de entre el vector entrante y 0, visible en la Figura 4.

$$G(x_i) = max(0, x_i) \quad donde \quad x_i \in \mathbb{R}^k$$

- **Linear:** función lineal que, como su nombre indica, devuelve el valor del vector entrante al cual se le aplica, ver en Figura 4. Es muy útil, como función de activación en capas de salida, en procesos relacionados con señales, dado que al contrario que las dos vistas anteriormente, no limita la salida a valores positivos.

$$G(x_i) = x_i \quad donde \quad x_i \in \mathbb{R}^k$$

- **Softmax:** función no-lineal similar en funcionamiento a la **sigmoid**. Es muy útil, como función de activación de salida, en problemas de clasificación con más de una clase, por ende, no es representable en dos dimensiones. Devuelve lo que podría considerarse la probabilidad a posteriori para cada clase presente en el problema.

$$G(x_i) = \frac{e^{x_i}}{\sum_{\forall i} e^{x_i}} \quad donde \quad x \in \mathbb{R}^k$$

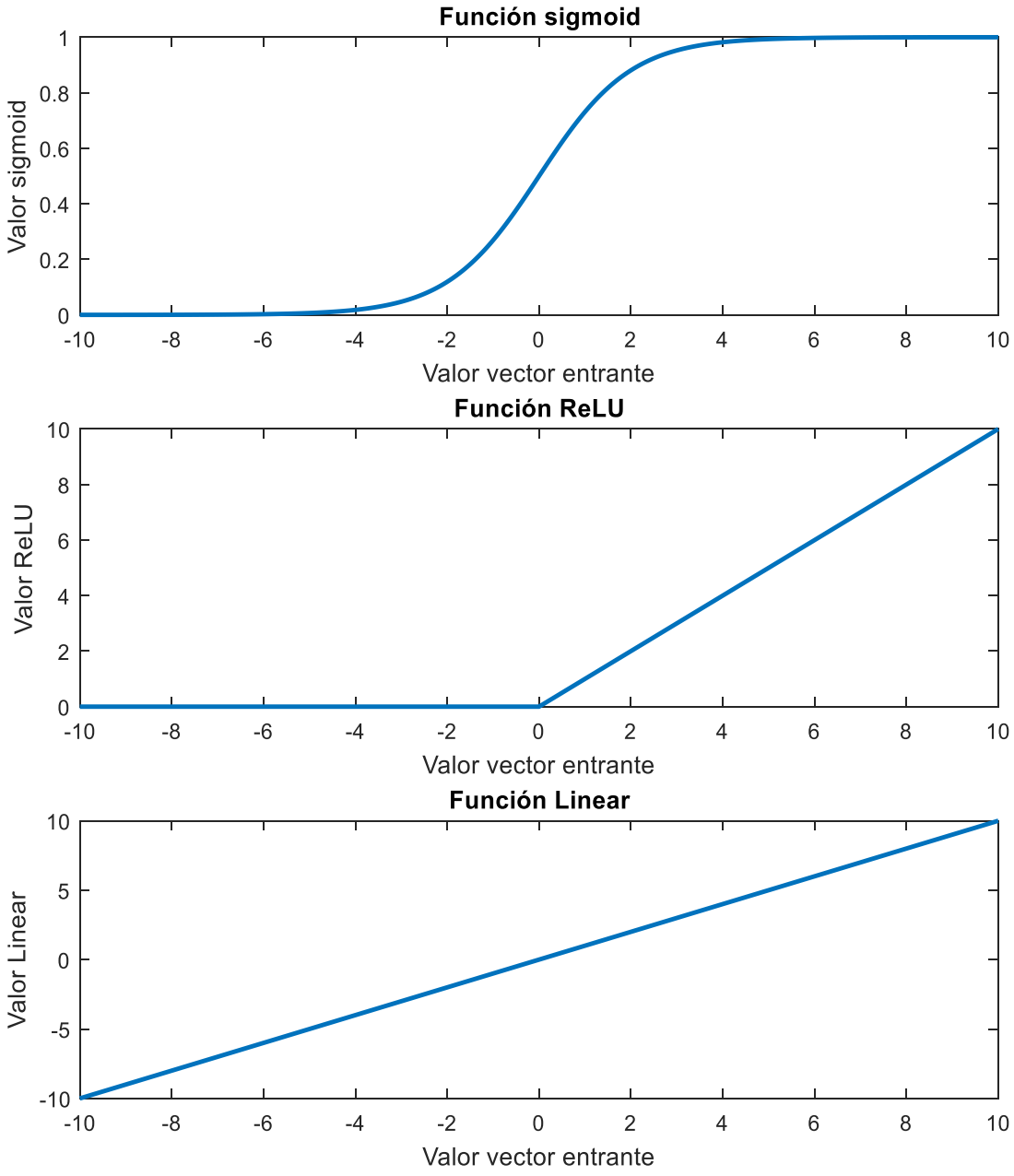

Figura 4: Representación de diferentes funciones de activación para un intervalo de valores de un vector entrante [-10, 10]

2.2.2 Función de coste y *backpropagation*

La función de coste, denominada C,es la última operación que se lleva a cabo en la red neuronal antes de realizar el *backpropagation* y pasar a una nueva iteración. Esta función define la manera en la que la red neuronal va a aprender y por ello es de vital importancia elegirla bien. De manera simple, la función de coste define de una forma matemática la relación que existe entre el resultado que la red consigue, la salida, y el que se querría que fuera, el objetivo o *target*, este resultado deberá ser, por lo general, minimizado. A este proceso se le denomina regresión y el principal objetivo será encontrar la posición mínima que pueda otorgarnos, es decir, que converja. Sin embargo, puede haber inconvenientes como que, por ejemplo, durante el entrenamiento la función de coste se estanque en un mínimo local, este no representa su mejor puntuación y por tanto habría que evitarlo, es ahí donde entran en juego los parámetros *learning rate* y *momentum* los cuales serán vistos en el siguiente apartado, así como otros métodos de regulación.

Como es de suponer, existen multitud de funciones de coste. A continuación, se exponen dos de las más usadas: [7]

- **MMSE (*Minimum Mean Square Error* o Error Cuadrático Medio Mínimo):** función muy usada en problemas de regresión, dado que lo que busca es que la función de salida de la red sea lo más parecida posible al objetivo o *target* planteado.

$$C = \frac{1}{2N}\sum_{n=1}^{N}(X'(n) - Y(n))^2$$

  Aquí N indica el número de dimensiones o muestras de cada vector, Y representa al vector de salida de la red neuronal y X' denota al vector objetivo.

- **Cros-entropía:** principalmente usada en problemas de clasificación de más de dos clases. Tiene una estructura diferente al **MMSE**. Se define y_k como las salidas de la red y x'_k como las clases target; en ambos casos el número total de clases se indica con K. Cabe destacar que, al ser la salida una probabilidad estas estarán en el rango [0,1]. [5]

$$C = -\sum_{k=1}^{K} x'_k * log(y_k) + (1 - x'_k) * log(1 - y_k)$$

Por otro lado, es muy importante mencionar la técnica *backpropagation*, esta es la encargada de 'realimentar' la red neuronal desde final a inicio con los datos obtenidos de la función de coste y mediante ellos actualizar los pesos en cada capa a fin de conseguir, por lo general, un menor coste en la siguiente iteración. Podría decirse que el objetivo de este algoritmo es el de minimizar el valor de la función de coste.

2.2.3 Matrices de pesos, vectores bias, tasa de aprendizaje y momentum

Las matrices de pesos, antes denominadas w^n, son estructuras de vital importancia en lo que al entrenamiento de una red neuronal se refiere. Los valores que guardan son aquellos que representan la combinación lineal existente entre las neuronas de una capa y cada una de las neuronas de la capa siguiente. Por ello la dimensión de una de estas matrices es la equivalente a relacionar dos capas consecutivas, es decir, si uno se encuentra en una capa de k dimensiones y la consecutiva tiene k', la matriz de pesos entre estas neuronas será $w^n \in \mathbb{R}^{k*k'}$.

Por otro lado, el vector bias, b^n, es una estructura constituida por una serie de constantes las cuales interactúan con cada neurona. Este vector tiene la misma dimensión que la capa a la que se atañe, es decir, si la capa de neuronas consta de 100 de estas, el vector bias de esta capa tendrá también 100 constantes. Estas constantes actúan como regulador del umbral que rige la función de activación de la neurona en sí. Por poner un ejemplo aclaratorio, si se tiene una función sigmoide, el umbral medio estaría en 0; sin embargo, si se introduce un bias de constante 1, dicho umbral se encontraría ahora en 1.

Es importante destacar que son estas estructuras las que iterativamente se van actualizando a fin de conseguir los valores óptimos para el problema que ocupe la red neuronal, es decir, de estos valores depende el buen funcionamiento final de la red.

Cabe destacar que evolucionan gracias a la función de coste y a la técnica de *backpropagation*, antes descritas. Como se ha dicho, se actualizan de manera iterativa, este proceso se lleva a cabo de manera correcta mediante un proceso matemático complejo que puede ser entendido gracias a la siguiente fórmula:

$$\Delta\left(w_{k+1}^{l}, b_{k+1}^{l}\right) = -\lambda \frac{\partial Er}{\partial\left(w_{k}^{l}, b_{k}^{l}\right)} - \kappa\, \lambda\left(w_{k}^{l}, b_{k}^{l}\right) + \omega\lambda\left(w_{k}^{l}, b_{k}^{l}\right)$$

$$siendo \quad 1 \leq l \leq L + 1$$

Donde L denota el número total de capas ocultas en la red neuronal y por tanto $L + 1$ representa a la capa de salida y l indica la capa en la que está aconteciendo la predicción. Por otro lado κ se refiere al coeficiente de decrecimiento de los pesos, en este TFG no se utiliza, y finalmente λ y ω constituye los denominados tasa de aprendizaje y *momentum*, respectivamente.

La tasa de aprendizaje o *learning rate* constituye un importante parámetro dentro del entrenamiento de las redes neuronales. De manera simplificada, permite limitar la rapidez de actualización de los pesos, es decir, hace que evolucionen más despacio y con ello evitar estancarse en mínimos locales de la función de coste. Por otro lado, el *momentum* es otro parámetro que cumple la misma función que el anterior mencionado. Ambos, son complementarios.

2.2.4 Batches y epochs

Como hemos visto anteriormente, la principal razón de ejecutar una red neuronal es la de minimizar la función de coste con cada iteración. De manera natural esta función se calcula para cada uno de los vectores de entrada y, por ende, se ejecuta el *backpropagation* para cada vector, es decir, para cada iteración; esto daría como resultado un entrenamiento muy preciso pero lento. Es por ello que surgió la idea de los batches o mini-batches. Estas estructuras no son más que bloques de vectores de entrada de la gran cantidad que suponen la base de datos. Ahora se calcula la función de coste por batch y no por entrada, con ello se acelera mucho el proceso de aprendizaje, si bien no resulta tan preciso como la manera inicial.

Es importante indicar que, si se elige un tamaño de batch muy grande, la función de coste se hará sobre mucha cantidad de datos y por lo tanto el entrenamiento no será el más óptimo; a este hecho se le denomina representación ruidosa de la función de coste.

Ahora que ya conocemos la estructura clave para un entrenamiento eficiente de la red neuronal, es hora de hablar de las denominadas *epochs* o épocas. Estas, al contrario que los *batches*, no son estructuras sino aglomeraciones de iteraciones. Se denomina *epoch* al hecho de haber ejecutado en la red todos los vectores de entrada de la base de datos, es decir, diremos que ha transcurrido una *epoch* cuando hayamos agotado los recursos de nuestros datos. Esto abre una posibilidad en lo que a entrenamiento se refiere, con este concepto un diseñador puede elegir el número de veces que desea entrenar la red con la misma base de datos; de alguna manera elige el tiempo de entrenamiento.

3. Diseño y desarrollo del TFG

A modo de guía se incluye un esquema, ver Figura 5, con las fases más importantes seguidas durante la implementación de este proyecto. Se comienza con la preparación de la señal de voz previa al entrenamiento, le sigue la creación del grafo y la elección de parámetros de la red neuronal y finalmente el proceso de reconstrucción utilizado para recuperar la señal.

Antes de comenzar con los diferentes bloques es importante hacer un inciso para hablar de las bases de datos utilizada. Para este proyecto se ha utilizado la base de datos TIMIT para voz, la cual está en inglés, y HU para ruido.

TIMIT fue diseñada para ofrecer datos de señales voz principalmente enfocados a la adquisición de conocimiento y al desarrollo de sistemas de automáticos de evaluación, limpieza o reconocimiento de señales de voz. Ofrece señales muestreadas a 16KHz de longitud variable y, además, sugiere una separación inteligente de señales en sets de entrenamiento y testeo. [8]

El bloque de entrenamiento consta de 630 locutores, hombres y mujeres, y 8 dialectos. Cada locutor consta de 10 frases habladas, esto hace un total de 6300 locuciones que ofrecen una gran variabilidad para sistemas como el que ocupa este TFG.

Por otro lado, el bloque de testeo puede diferenciarse en dos distintos en función de las necesidades. El denominado '*Core Test Set*', el cual ofrece 24 locutores y 8 dialectos distintos; por cada dialecto se diferencian dos locutores varones y una locutora mujer, dando como resultado un total de 192 señales de voz con las que interactuar. El otro bloque recibe el nombre de '*Complete Test Set*' y expande las capacidades del bloque anterior; consta de 168 locutores y los mismos 8 dialectos, todo esto da como resultado un total de 624 frases habladas. Cabe destacar que es este último bloque el que se ha utilizado para la realización de pruebas en este trabajo.

De HU tan solo comentar que consta de 100 tipos de ruidos diferentes. Gracias a su cantidad se dará una gran variabilidad a las señales utilizadas para el entrenamiento.

3.1 Preparación previa al entrenamiento de la señal de voz

En este apartado se explicarán las transformaciones que se llevan a cabo en la señal de voz que se pretende limpiar para que, una vez finalizado, la señal esté preparada para ser introducida en la red neuronal entrenada a fin de que nos la devuelva con ruido reducido.

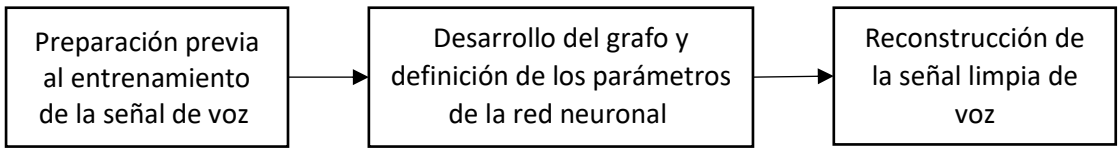

Figura 5: Esquema de las fases más relevantes del proyecto.

3.1.1 Enventanado y cálculo del espectro

En este bloque se lleva a cabo un enventanado de la señal de voz con ventanas Hamming de N=512 muestras o su equivalente en tiempo 32ms, estas ventanas deberán tener un solapamiento a elección del diseñador. En audio típicamente el solapamiento es de la mitad del ancho de una ventana, en el caso de este TFG el solapamiento es del 50% con M=256 muestras o 16ms. Una forma sencilla de calcular una ventana Hamming es:

$$w[n] = c(\alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right)) \qquad 0 \leq n \leq N-1$$

$$Siendo\ \alpha = 0.54, \beta = 0.46$$

El reto en este punto es decidir que ganancia *c* es conveniente en la ventana Hamming para conseguir que la suma de todos los frames solapados en tiempo sumen 1. Esto es importante ya que, de no ser así, no será posible una reconstrucción óptima en tiempo al finalizar el trabajo, se comprobó que una reconstrucción simple con una c distinta de 1/1.08, ver Figura 6, provocaba que el MSE entre la señal recuperada y la original fuera demasiado elevado.

Una vez aclarado lo anterior, será el momento de calcular la DFT (*Discrete Fourier Transform* o Transformada Discreta de Fourier) $X[k]$ de cada uno de los intervalos eventanados $x[n]$ de la señal de voz.

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{nk} \qquad Siendo \qquad W_N^{nk} = e^{-j(\frac{2\pi}{N})}$$

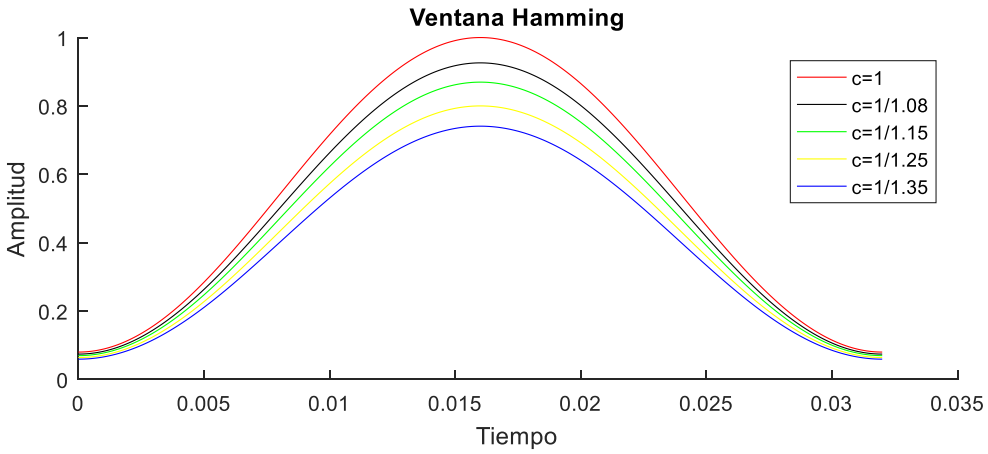

Figura 6: Comparativa de ventanas Hamming para diferentes ganancias c

3.1.2 Acondicionamiento y guardado de los datos

Los resultados de cada DFT deberán ser guardados en matrices; concretamente dos, una para los módulos de los espectros y otra para las fases de los mismos. Cada una de estas matrices tendrá una cantidad de filas equivalente al número total de ventanas extraídas de la señal de voz y un número de columnas equivalente a la mitad del tamaño de las DFTs más una unidad, en este caso 257 muestras. Este cambio de longitud en los datos se debe a la simetría que sufre el espectro de una señal, la cual solo aporta información redundante.

Una vez que se han establecido los tamaños de las matrices, se guardará en una de ellas, denominada matriz F, por filas las fases de cada ventana de 257 muestras resultante de la DFT. Esta permanecerá sin modificar puesto que no será necesaria de nuevo hasta la reconstrucción final de la señal.

Por otra parte, en la matriz restante, denominada matriz M, se guardará la energía $E[k]$ del módulo de la mitad del espectro de la DFT de las diferentes ventanas. Este resultado es fácilmente calculable mediante la siguiente formulación:

$$E[k] = 20\log_{10}(|X[k]|) \qquad 0 \leq k \leq N/2$$

Ahora se puede conseguir la representación del espectrograma, ejemplo en la Figura 7, de la señal de una manera sencilla. Esto aporta información rápida del estado espectral de la señal. Es decir, se puede hacer una primera aproximación a la cantidad de ruido, de haberlo, que presenta la señal que se quiere tratar.

Una vez se tienen los resultados, es momento de prepararlos para que sean compatibles con una red neuronal y puedan ser válidos. Este proceso es simple, consiste en dar lo que se denomina *entorno* a cada una de las filas o frames guardados en la matriz M. Esto da contexto a la red neuronal más allá del frame que se pretende limpiar. En el caso de este TFG se ha elegido un entorno de diez frames, es decir, cinco frames antes y cinco después del instante temporal entrante en la red neuronal, con ello conseguimos un total de 257*11=2827 muestras por cada vector de entrada a la red neuronal; esta nueva matriz, denominada matriz E, deberá ser guardada independientemente de la anterior. Para evitar problemas, se eligió introducir cinco frames nulos, de tamaño igual a los ya guardados en la matriz de módulos M, al principio y al final de la misma. Con esta acción se evita el incoveniente de quedarse sin señal para los frames tanto iniciales como finales.

3.2 Desarrollo del grafo y definición de los parámetros de la red neuronal

En este apartado se explicarán los pasos llevados a cabo para la generación de un grafo óptimo para la tarea que nos ocupa.

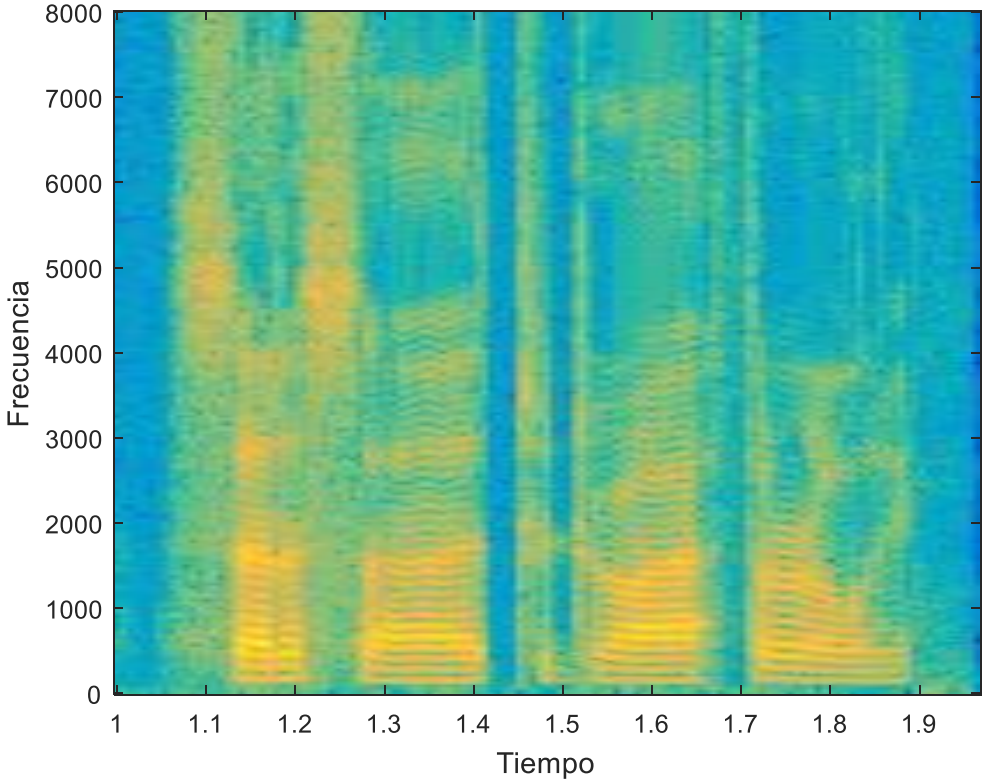

Figura 7: Ejemplo de espectrograma de un audio limpio

3.2.1 Elección de tamaño de grafo

Para el caso concreto de este TFG se ha implementado una red neuronal *fully connected*, esto es un grafo donde todas las neuronas están totalmente conectadas entre capas, de 2857 dimensiones de entrada, esto concuerda con el número de muestras que quedaron almacenadas por fila en la matriz E que se mencionó anteriormente. De este número de entradas se pasa a la primera capa oculta, de tres que tiene el grafo. Esta capa consta de 2048 neuronas, al igual que las demás. Para finalizar el sistema hay una capa de salida de 257 dimensiones, estas concuerdan con el número de muestras que tiene cada ventana guardada en la matriz M, es a este tamaño al que queremos aspirar pues es el necesario para reconstruir apropiadamente la señal de voz con el problema actual. En definitiva, el grafo se entiende muy fácilmente con la Figura 8, en ella se ha ilustrado de la manera más visual posible toda la distribución de la red.

3.2.2 Elección de vectores de bias y matrices de pesos

Estas estructuras, como se vio anteriormente en el apartado 2.2.3, son clave para poder guardar los datos de un entrenamiento y así poder llevar a cabo limpiezas muy rápidas sin necesidad de reentrenar la red neuronal cada vez que se quiera utilizar.

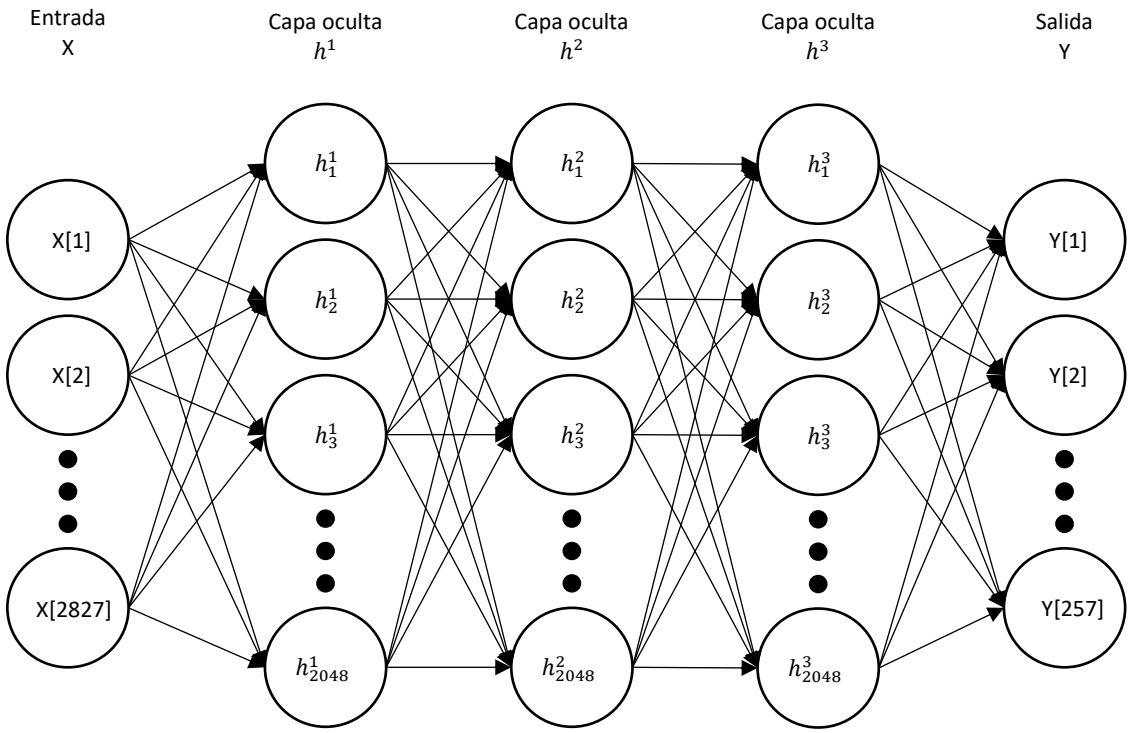

Figura 8: Ilustración del grafo implementado en el TFG

Las matrices de pesos, denominadas w^n, son estructuras de tantas filas como la capa de la que vengan y tantas columnas como la capan a la que vayan, y son tan numerosas como intersecciones haya entre capas, para este TFG corresponden cuatro matrices w^n. Por ejemplo, si estamos en la capa de entrada y hay que llegar a la primera capa oculta, se tendrá una matriz de pesos w^1 de dimensión 2827x2048.

Por otro lado, es importante destacar a los vectores de bias, b^n. Estos se caracterizan por existir uno por intersección de capas y tener tantos puntos como número de elementos tenga la capa hacia la que se dirija la intersección. Por ejemplo, si de nuevo nos encontramos entre la capa de entrada y la primera capa oculta, el vector b^1 tendrá una dimensión de 2048x1.

Todos estos vectores y matrices se inicializan a cero y de manera aleatoria con distribución Gaussiana de media 0 y varianza 1, respectivamente. Con el paso de las iteraciones cambiarán sus valores y, al finalizar el entrenamiento, es con estos valores con los que deberemos quedarnos a fin de poder utilizar la red a modo de test.

Por último, se tiene la matriz de resultados. Esta matriz, denominada matriz L, deberá tener el mismo tamaño que la matriz de la energía de los módulos de la señal enventanada que se mencionó en el punto anterior, la matriz M. Para este problema, la matriz tendrá tantas filas como ventanas tuviera la señal a limpiar y tantas columnas como muestras tenga cada ventana, en este TFG esta cifra deberá ser 257.

3.2.3 Funciones de activación y función de coste

Llegados a este punto es el momento de elegir los mecanismos de aprendizaje de la red neuronal. Las claves de este problema son, sin duda, las funciones de activación de las capas ocultas y la función de coste. Para las neuronas de las capas intermedias, sin incluir la capa de salida, se ha optado por una función de activación del tipo ReLU (*Rectified Linear Unit*). Como se explicó en el apartado 2.2.1, esta función elige el máximo entre el número que se calcule en la neurona correspondiente y cero.

$$f = \max(0, x)$$

$$Siendo\ x\ el\ valor\ calculado\ en\ la\ neurona$$

Por otro lado, en la última capa, la de salida, se optó por una función de activación del tipo linear. Esto es que lo que se calcula en cada elemento de salida, sin modificar, es lo que se considera salida de la red neuronal. Se implementó así debido a que, de seguir usando ReLU, jamás se conseguirían valores negativos y, es obvio, los espectros de amplitud en unidades logarítmicas tienen muchos.

$$f = x$$

$$Siendo\ x\ el\ valor\ calculado\ en\ la\ neurona$$

Como última función de la red neuronal, es muy importante destacar el papel que juega la función de coste, para este TFG se usó el MSE. Esta se eligió porque la prima máxima al entrenar la red neuronal era que los frames que se van extrayendo de la misma en el entrenamiento se parecieran lo más posible al *target*, en este caso el frame original limpio. Se intenta reducir este error iterativamente con un descenso por gradiente, lo cual es lo más lógico dadas las circunstancias.

3.2.4 Elección del tamaño de batch y número de epochs

Es ahora cuando toca elegir el tamaño del batch. Como se explicó en el apartado 2.2.4, un batch es un elemento que se define como un número concreto de señales de entrada de la base de datos a la red, es decir, como paquetes de datos dentro del global de la base. En el caso de este TFG, estas señales serán las filas de la matriz E antes mencionada. Así pues, el tamaño de batch para este proyecto es de 100 pues interactúa muy bien con el número total de vectores de entrada que se tiene como base de datos.

Una vez se ha llegado a este punto, simplemente se ha de elegir el tiempo de entrenamiento que se quiere emplear. Este tiempo se escoge, no eligiendo unos minutos o horas y ejecutando, sino eligiendo el número de iteraciones totales que quieren realizarse. A esto se le denomina epoch, ya visto en el apartado 2.2.4. A modo de recordatorio decir que una epoch es una iteración completa a toda la base de entrenamiento, es decir, cuando se ponga en ejecución el entrenamiento de la red neuronal se dirá que ha transcurrido una epoch cuando todos los datos de entrenamiento hayan pasado por la red una vez. Como consecuencia cuantas más epochs más tiempo de ejecución; sin embargo, habrá que tener cuidado con no sobreentrenar la red con los datos de entrenamiento o se correrá el riesgo de hacer que esta funcione muy bien para los datos de entrenamiento, pero no para los de test. Para este TFG el número de epochs no se ha fijado puesto que constituye uno de los parámetros clave de las pruebas realizadas, como se verá más adelante.

Cabe destacar, aunque en este proyecto no se haya implementado, que existen entrenamientos más elaborados de las redes neuronales, que permiten terminar la ejecución del entrenamiento no cuando hayan pasado un número determinado de epochs, sino cuando se cumpla una determinada condición; por ejemplo, que la función de coste se haya mantenido más o menos estable durante un número razonable, a elección del diseñador, de batches.

3.2.5 Implementación y entrenamiento la red neuronal

En este proyecto se utilizó Theano, más concretamente Blocks, para diseñar y entrenar la red neuronal. Gracias a estas librerías se consigue de una manera relativamente sencilla, establecer cada uno de los parámetros vistos durante este bloque. Una vez se ha decidido el tamaño que tendrá la red, es decir, el tamaño de las matrices w^n y de los vectores b^n, el usuario solo deberá generar con variables *shared* (variables propias de Theano) las estructuras correspondientes, implementando en el proceso las funciones de activación y finalmente la de coste.

Una vez hecho esto solo se necesitará abrir la base de datos, base previamente estructurada para que Blocks sea capaz de leer, y ejecutar la red con el tamaño de batch y número de epochs requeridos.

Sin embargo, surgió un inconveniente. La red funcionaba como se esperaba hasta un máximo de tamaño de base de datos. Si se intentaba implementar una base de datos de un tamaño mayor al que equivalen 10 horas, el sistema no tenía suficiente memoria RAM para mantener las considerables matrices. Por ello se tuvo que idear una manera de entrenar la red con grandes cantidades de datos sin que ello supusiera un inconveniente.

Finalmente, se decidió implementar una nueva red que, en vez de entrenar con una base completa, entrenara con bases más pequeñas de tamaño fijo. Se llegó a la conclusión de que, si se necesitaba entrenar una red durante 50 epochs con una base de datos de 50 horas, lo que se podía implementar era una red que entrenara en un inicio con una base de 10 horas durante tan solo un epoch. Al finalizar este proceso se ejecutaría otra red con las

matrices w^n y vectores b^n inicializados con los valores de entrenamiento que la anterior red proporcionó, y así sucesivamente hasta realizar un entrenamiento de una epoch para cada una de las bases de datos de 10 horas. Una vez se ha hecho una epoch para todas las bases, tan solo se necesitará repetir el proceso durante el número de epochs que el sistema requiera en un inicio, en este ejemplo 50 veces. Finalmente se guardarán los valores de los vectores b^n y las matrices w^n finales y será con estos con los que se implementen los tests posteriores.

3.3 Reconstrucción de la señal limpia

En este bloque se va a explicar todo el proceso necesario que hay que seguir para, una vez se ha conseguido la matriz de resultados L, reconstruir de la mejor manera posible la señal de voz para poder escucharla y, si cabe, realizar pruebas de calidad en la limpieza de la red.

Todo empieza con la duplicación de las columnas de L, esto se hace porque en la preparación previa se recortó a la mitad el espectro debido a su redundancia. Para este paso es imprescindible recuperar esa información o no se podrá realizar la DFT inversa.

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\, W_N^{-nk} \qquad Siendo \qquad W_N^{-nk} = e^{j(\frac{2\pi}{N})}$$

Una vez se tiene la DFT inversa, habrá que reconstruir la señal en tiempo, para ello se recogerán las ventanas recuperadas y se reestructurarán en forma de vector, perdiendo así la forma de matriz. Hay que tener cuidado para realizar correctamente el solapamiento. En la fase previa al entrenamiento se llevó a cabo un solapamiento del 50%, esto ahora hay que tenerlo en cuenta. Simplemente se deberá sumar cada 50% inicial de cada ventana con el 50% final de la ventana anterior.

Cabe destacar que la matriz que la red neuronal devuelve es una matriz de energías, así pues, antes de llevar a cabo cualquier DFT inversa, deberemos recuperar el espectro en amplitud no logarítmica.

3.4 PESQ, técnica de valoración

Este método de valoración objetiva, estandarizado por la ITU y actualmente muy utilizado a nivel mundial en la industria, tiene como fundamento principal comparar una señal $x[n]$ con una señal degradada $y[n]$ obtenida a partir de la propia señal $x[n]$; para el caso de este TFG la degradación se aplica añadiendo ruido con la herramienta FANT desarrollada

en el proyecto AURORA. Así pues, la salida de PESQ será una predicción de calidad percibida por "sujetos" en una escucha subjetiva atribuida a $y[n]$.

En primera instancia, PESQ calcula unos retardos, de ser suficientemente significativos en el intervalo de tiempo elegido, entre $x[n]$ e $y[n]$. Para cada uno de ellos se calcula el lugar de inicio y de parada. Una vez en este punto se utiliza un algoritmo de alineación sobre $y[n]$ que compara el nivel de confianza que se obtiene cuando hay dos retardos en un intervalo de tiempo con el que se consigue en un intervalo con un solo retardo. Llegados a este punto, sobre esta base de retardos, PESQ compara $x[n]$ y la señal $y[n]$ alienada, denominada $y_a[n]$, mediante un modelo de percepción como el de la Figura 9. En este proceso se hace una transformación de las señales a tratar en una representación interna, esta es una representación análoga a la representación psicofísica de las señales auditivas en un sistema de recepción humano.

La representación interna de estas señales se procesa para poder tener en cuenta efectos del tipo: variaciones de ganancia local o filtrados locales. Estos efectos, si no son muy grandes, puede que no tengan una gran repercusión en el sistema y, por ende, no supongan una gran influencia en lo que a percepción se refiere.

Finalmente, PESQ nos devolverá dos parámetros fundamentales, el Raw-MOS y el MOS-LQO. Estos valores serán los que se pueden utilizar para comparar y hacer valoraciones entre los audios ruidosos y los que han sido extraídos de la red neuronal, supuestamente limpios. Cabe destacar que estos valores tienen un valor máximo igual a 5 y que cuanto

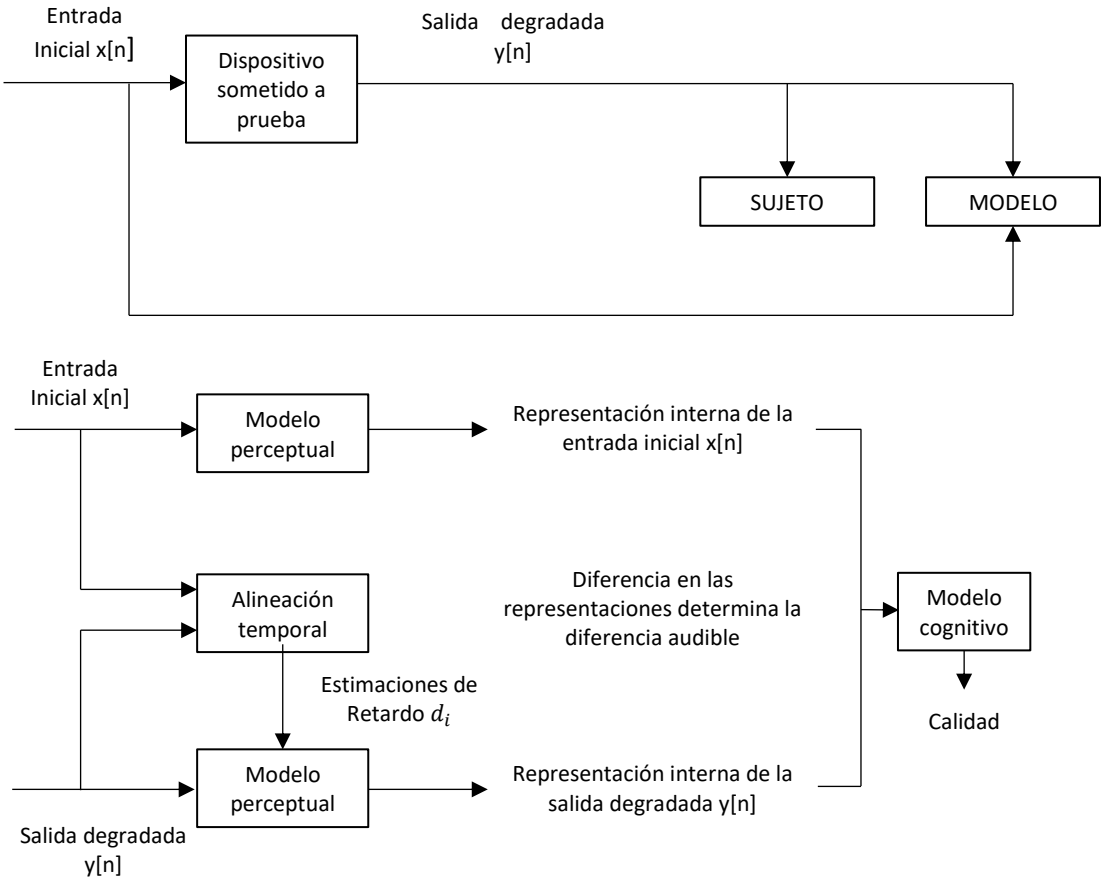

Figura 9: Modelo de percepción utilizado por PESQ

más alto sea el valor devuelto por PESQ más similar será la señal comparada con la original $x[n]$. También añadir que para este proyecto solo se ha utilizado el valor Raw-MOS para tener una comparación directa y coherente con los resultados mostrados en '*A Regression Approach to Speech Enhancement Based on Deep Neural Networks*', estudio que, como se dijo antes, se ha utilizado como referencia.

Por último indicar que, en este proyecto, se han realizado dos comparaciones fundamentales, se implementó PESQ para comparar los audios originales con aquellos a los que se le introdujo ruido y posteriormente se hizo lo mismo, pero comparando los originales con aquellos devueltos por la red neuronal.

4. Pruebas y resultados

A lo largo de este apartado se van a exponer los diferentes resultados PESQ obtenidos en las distintas pruebas que se han llevado a cabo. Cada una de las pruebas pretende sacar a la luz los diferentes comportamientos que se pueden obtener de la red neuronal en función de: tamaño de base de datos, tiempo de entrenamiento, limpieza con ruidos independientes al entrenamiento y limpieza con señales de distinto idioma de test.

4.1 Resultados PESQ para diferentes tamaños de base de datos y 50 epochs

En este bloque se reflexionará sobre los resultados obtenidos tras el procesado de los audios de la base de test TIMIT, la *Core Test Set* mencionada al principio del apartado 3, la cual consta con 624 audios diferentes; dichos audios son tratados con los mismos ruidos usados durante el entrenamiento (base de datos de ruidos HU), estos se añaden a las señales de audio escogiendo uno de manera aleatoria para cada señal. Se han realizado pruebas con diferentes SNR para cada tamaño de base de datos de entrada de la red neuronal, en cada una de ellas se han utilizado la totalidad de los audios disponibles.

Como se puede apreciar en la Tabla 1, los resultados evaluados mediante PESQ, visto en el apartado 3.4, para los audios ruidosos sin tratar son, por lo general, los esperables. El valor del MOS asciende según asciende el SNR tratado; sin embargo, durante las pruebas se observó que ocurrían ciertas irregularidades. Como se puede ver en la columna referente a los audios de 10 dB de SNR, el resultado de media arrojado por PESQ, resultan desconcertantes dado que son menores o mayores, respectivamente, a los obtenidos para un valor de SNR igual a 5 dB. A pesar de que se han repetido las prueba en numerosas ocasiones y se han comprobado los resultados en busca del origen de esa anomalía, todavía no se ha encontrado a qué se debe.

Como cabe esperar, el objetivo de este proyecto es que el resultado al utilizar PESQ, en cada audio al salir de la red neuronal, mejore lo más posible los resultados vistos en la Tabla 1. Resulta evidente, tras mirar la Tabla 2, que los resultados de MOS son claramente mejores tras la limpieza en la red. De nuevo es evidente que la red mejora sus resultados cuanto más grande sea la base de datos con la que se trabaja. Sin embargo, es destacable el hecho de que, para la red de 1h de base de datos, los resultados son muy parecidos a los de la Tabla 1 o incluso peores.

	SNR				
Audios Sucios	-5	0	5	10	15
Media de MOS	1,3633	1,5754	1,8043	1,6646	2,3151

Tabla 1: Resultados de media de PESQ obtenidos para la base de datos de test de audios ruidosos sin mejorar.

Media de MOS	SNR				
Horas de Base de Datos	-5	0	5	10	15
1h	1,3953	1,6203	1,8219	1,6231	2,1211
2h	1,5484	1,7774	1,9755	1,7344	2,2777
5h	1,7773	2,0054	2,1916	1,8857	2,4786
10h	1,7293	1,9961	2,2126	1,8729	2,5277
20h	1,8583	2,1187	2,3243	1,951	2,6328
50h	1,9894	2,2363	2,4356	1,9979	2,7414
90h	2,0191	2,277	2,4808	2,0212	2,7853

Tabla 2: Resultados de media de PESQ obtenidos para diferentes tamaños de bases de datos y mismo número de epochs de entrenamiento.

También es remarcable que las diferencias existentes entre los resultados de 50h de base de datos y los de 90h no son muy grandes, logrando en ambos casos unos valores muy parecidos. Por ejemplo, para una señal con 5 dB de SNR y 50h de base de datos se obtiene

un valor medio de 2,4356 en el PESQ y para el mismo SNR y 90h de base se obtiene un 2,4808. Aunque esto pueda hacer preguntarse si realmente merece la pena el tiempo y los recursos de más que requiere entrenar una red neuronal, de cerca del doble de datos, para un escaso 2% de mejora, hay que tener en cuenta que el coste temporal se da solo durante

el entrenamiento y que una vez entrenada la red los costes computacional y temporal al ejecutarla son escasos y siempre similares, de modo que incluso esta escasa mejora claramente merece la pena.

Por otro lado, y como era de suponer, la irregularidad presente en los casos de 10 dB de SNR se mantiene en los resultados obtenidos tras ejecutar la red. Como se muestra en la Figura 12, la distribución para los casos de 10 dB de SNR siguen una estructura similar a dos gaussianas adosadas, mientras que los demás casos, mostrando un ejemplo para el caso de audios con 5 dB de SNR en la Figura 11, siguen una distribución similar a una gaussiana pura; son estas últimas las esperables pese a no conseguirlas en el caso de 10 dB de SNR.

De una manera más visual se puede apreciar en la Figura 10 la gran mejora que supone utilizar la red. Se corrobora que los espectros de los audios ruidosos sufren una limpieza significativa en cuanto son tratados; todo ello da como resultado, en todos los casos, espectros mucho más parecidos al original.

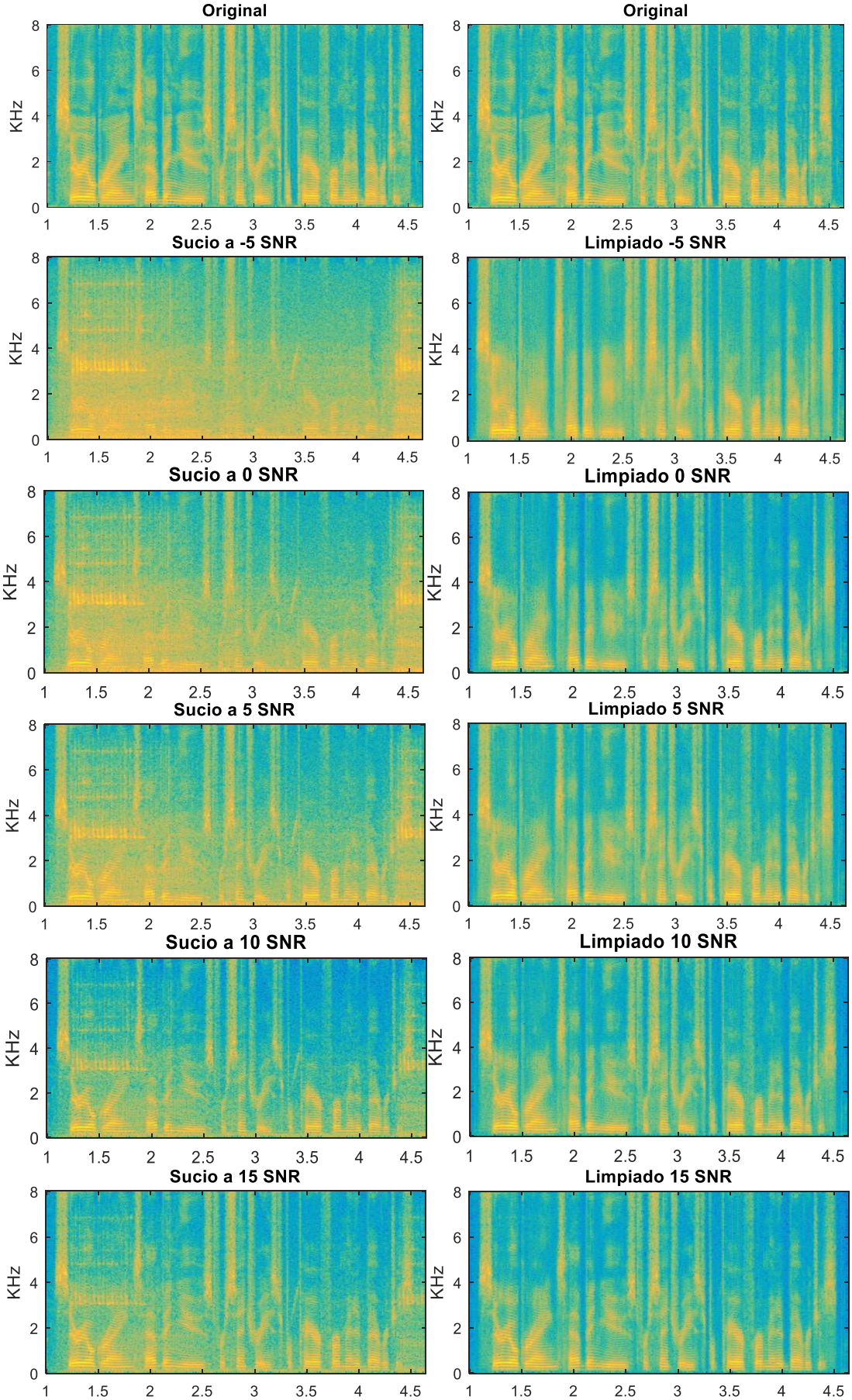

Figura 10: Ejemplos de espectrogramas sucios (izquierda) y limpiados con una red de 90h de base de datos y 50 epochs (derecha) para un mismo audio original (arriba)

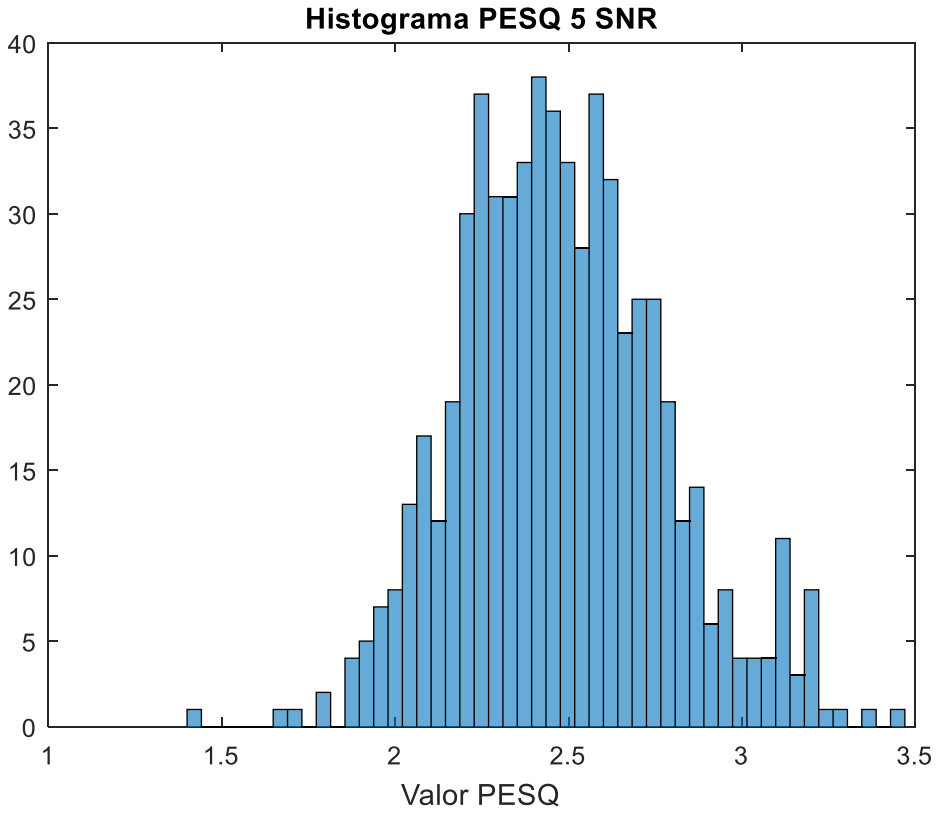

Figura 11: Ejemplo de la distribución gaussiana de valores PESQ para un caso de SNR = 5

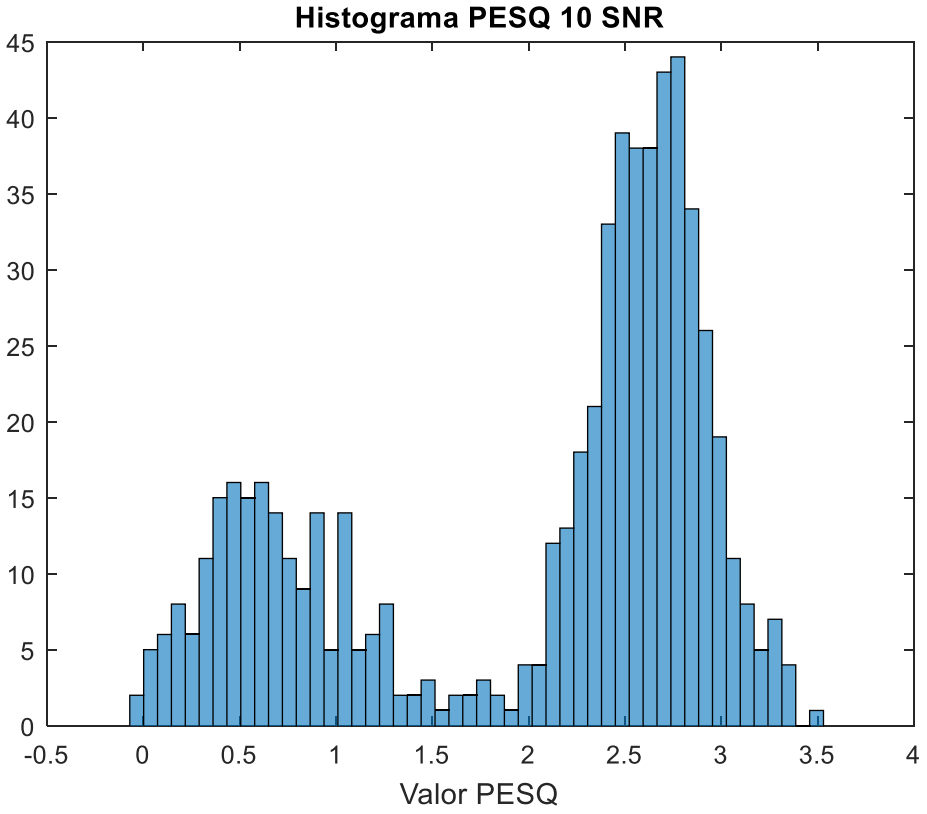

Figura 12: Ejemplo del malfuncionamiento en distribución del PESQ en el sistema para señales de SNR = 10

4.2 Resultados PESQ para diferentes epochs y 90h de base de datos

En estas pruebas se ha querido estudiar la repercusión que tiene el entrenar durante más o menos tiempo, en este caso epochs, la red. Para ello se realizaron una batería de pruebas de 10 a 100 epochs y como en los casos anteriores se ejecutó PESQ para cinco SNR distintos. Al igual que en el aparado anterior se usó la totalidad de los audios del *Core Test Set* de TIMIT para cada prueba, con los mismos ruidos HU de nuevo escogiendo uno de manera aleatoria para cada señal de voz.

A la vista de los resultados, ver Tabla 3, se comprueba que si bien mejora cuantas más epochs se entrena, esta mejora no es tan drástica como se vio en el apartado anterior para los diferentes tamaños de bases de datos. Esto hace pensar que, si bien el número de horas de base de datos es un punto clave a la hora de dar variabilidad a los datos de entrenamiento, no lo es tanto el tiempo requerido para el entrenamiento.

Si tomamos como ejemplo el caso de 50 epochs, número de epochs empleado en el apartado anterior para todos los entrenamientos, se ve que para 15 dB de SNR se consigue un MOS de 2,7853. Por otra parte, si tomamos el mismo valor de SNR, pero para un número superior de epochs, en este caso 100 epochs, se aprecia que el valor que devuelve PESQ es de 2.8265. Al igual que ocurría en el apartado anterior, la mejora es muy pequeña, apenas un 1,5%; de nuevo es planteable si merece la pena o no el doble de tiempo de entrenamiento para una mejora tan escasa. Sin embargo, y pese a todo lo anterior comentado, tal como se explicó en el apartado anterior un aumento en el tiempo de entrenamiento de la red, por muy drástico que sea, no tiene impacto real en el coste computacional y temporal al aplicar dicha red una vez entrenada, por lo que se puede concluir que sí merece la pena dado que los resultados serán mejores en cualquier caso.

Cabe mencionar que los casos atípicos, ocurridos para un SNR de 10 en el apartado anterior, se repiten para estas pruebas y por ello volvemos a encontrar peores resultados en este SNR que en cualquier otro exceptuando el valor -5. La distribución resultante es muy similar a la vista en el apartado anterior, se vuelve a tener una doble gaussiana como la de la Figura 12 para los casos anómalos de 10 SNR y la distribución gaussiana habitual para los demás casos, como se aprecia en la Figura 11.

En la Figura 13, se pueden ver algunos ejemplos de espectrogramas para un SNR fijo y diferentes epochs de entrenamiento. Se aprecia que, como se vio en la Tabla 3, las diferencias, para los espectrogramas de las señales tras aplicarles la red neuronal, son muy escasas y de hecho casi inapreciables visualmente.

Raw MOS/Media	SNR				
Epochs de entrenamiento	-5	0	5	10	15
10 epochs	1,8633	2,1259	2,3387	1,9621	2,6642
20 epochs	1,9504	2,212	2,4154	1,9906	2,7256
50 epochs	2,0191	2,277	2,4808	2,0212	2,7853
100 epochs	2,0581	2,3161	2,5214	2,0384	2,8265

Tabla 3 Resultados de media de PESQ para una base de datos de 90h y una variación de las epochs de entrenamiento.

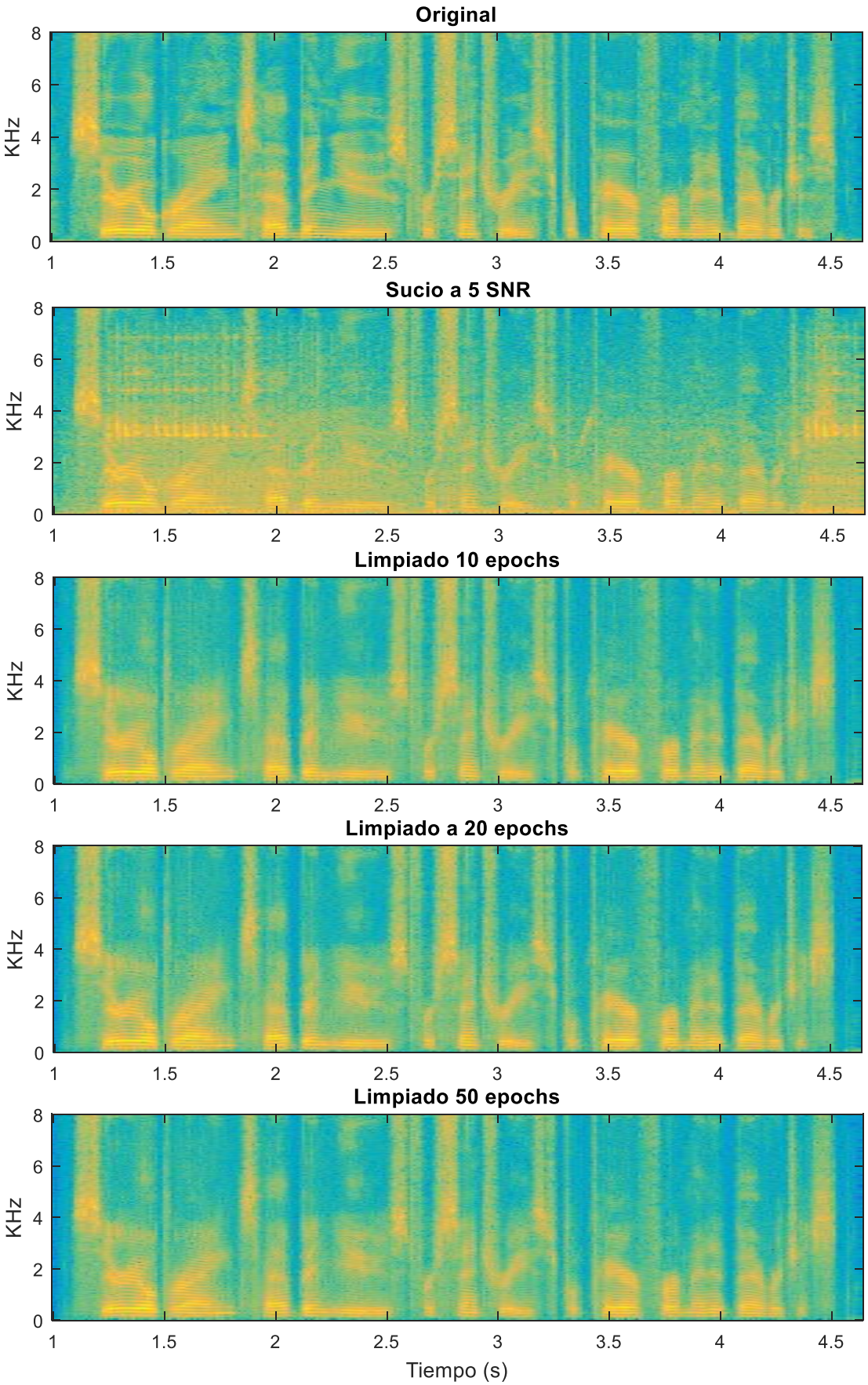

Figura 13: Ejemplos espectrogramas: audio original (arriba), audio ruidoso original (debajo del anterior) y audios limpiados con 90h y diferentes epochs.

4.3 Resultados PESQ para ruidos NOISEX

En este bloque se pretende comprobar el rendimiento que ofrece una red neuronal entrenada para la base de datos de ruidos HU empleada hasta el momento, con una base de test ruidosa con ruidos que la red no ha visto nunca anteriormente. Como audios se vuelven a usar la totalidad de la *Core Test Set* de TIMIT en cada prueba. Los ruidos empleados en este apartado para el test son los pertenecientes a la base de datos NOISEX, esta proporciona un total de 14 señales ruidosas de origen militar y por lo tanto ofrecen un contexto muy diferente a los vistos hasta ahora. Al igual que en apartados anteriores, se escoge uno de estos ruidos de manera aleatoria para cada señal de voz.

En la Tabla 4 se proporcionan los resultados PESQ para las señales ruidosas que se pretenden limpiar. Se observa que se parte de unos valores muy similares a los que se presentaron en la Tabla1.

Por otro lado, también en la Tabla 4, se exponen los resultados de media para los audios extraídos de la red neuronal para dos cantidades de epochs diferentes. Se ha optado por utilizar la red entrenada de 90h de base de datos, ya que es la que mejor rendimiento ofrecía en apartados anteriores. Se comprueba que, si bien los resultados no son tan buenos como para los ruidos de los apartados anteriores, sí que se aprecia una mejora consistente al aplicar la red neuronal.

Así pues, se observa que para, por ejemplo, valores de SNR de 15 dB, el MOS obtenido de PESQ es de 2.5896 en el caso de 50 epochs y de 2.6099 en el caso de 100 epochs. Este hecho dice que el rendimiento obtenido para ambas redes es apenas es un 7% menor, para la red de 50 epochs, y de un 6,3% menor, para la de 100 epochs, que sus redes equivalentes para ruidos de la base de datos HU de los apartados anteriores. Esto quiere decir que la red neuronal responde de manera satisfactoria antes situaciones, a priori, adversas. En la Figura 14 se pueden observar algunos espectrogramas para una evaluación más visual. Con esta evaluación se aprecia que la red es capaz de limpiar considerablemente el espectro, incluso cuando no ha sido entrenada con el mismo tipo de ruido que aparece en el audio a limpiar.

Algo destacable de estas pruebas es que, si en las dos anteriores había irregularidades con los audios a 10 dB de SNR, en estas no se han dado. Con ello se puede pensar que estos casos anómalos provienen de la base de datos utilizada de ruidos, HU, utilizada hasta el apartado anterior.

	SNR				
	-5	0	5	10	15
Media de MOS Audios Sucios NOISEX	1,2567	1,4632	1,7239	1,9934	2,2519
Media de MOS Audios Limpios NOISEX 50 epochs / 90h	1,597	1,8932	2,1673	2,3936	2,5896
Media de MOS Audios Limpios NOISEX 100 epochs / 90h	1,5936	1,8943	2,1713	2,4057	2,6099

Tabla 4: Resultados de media del PESQ para audios ruidosos con NOISEX, para audios limpiados con red de 90h y 50 o 100 epochs.

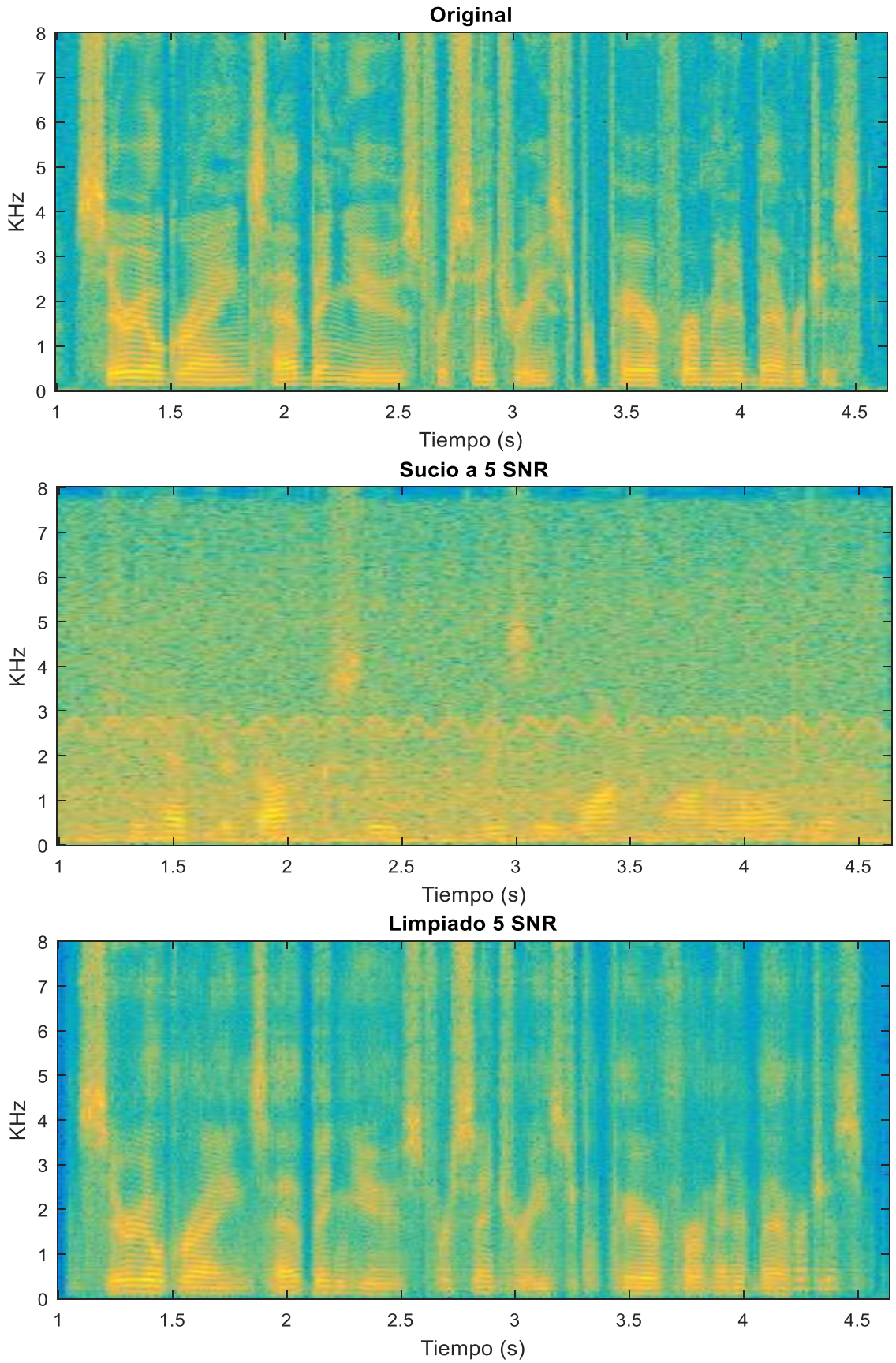

Figura 14: Ejemplo de Espectrogramas con base de datos de ruidos NOISEX: original (arriba), ruidoso original (medio) y procesado por la red neuronal (abajo).

4.4 Resultados PESQ para base de datos de voz ALBAYZIN sin cambio de ruidos

Con estas pruebas se quiso estudiar el comportamiento de la red neuronal ante otra situación adversa, en este caso la utilización de una base de datos de test en un idioma diferente al de entrenamiento. Para ello se utilizaron en cada prueba 500 ficheros de la base de datos de test de ALBAYZIN, esta está constituida por señales en castellano, interpretadas tanto por hombres como por mujeres. Se volvió a usar la base de datos de ruidos HU, siendo estos añadidos a las señales de voz de la misma manera que en los apartados anteriores.

En la Tabla 5 se presentan los resultados MOS devueltos por PESQ para los audios ruidosos originales. Si bien los resultados para SNR bajos son algo menores que los vistos hasta el momento, es destacable que para SNR altos los valores devueltos son mucho mayores que los vistos anteriormente.

De nuevo, en la Tabla 5 se ofrecen los valores MOS de PESQ para la red de 90h de base de datos y 50 epochs de entrenamiento. Curiosamente se ve que la limpieza de la base de datos en castellano devuelve valores PESQ muy superiores a los vistos anteriormente con la base de datos TIMIT. Resultan, pues, sorprendentes estos resultados. Como se ha comentado anteriormente, la red neuronal ha sido entrenada con la base de datos TIMIT y responde de manera muy superior para señales ruidosas de diferente idioma, concretamente la mejora es del 14% con respecto al mejor resultado obtenido en el apartado 4.1.

Al igual que en las pruebas anteriores, en la Figura 15, se ofrecen resultados visuales a modo de espectrogramas de alguna de las señales procesadas. Se puede observar la mejora en el espectro que se consigue con la red neuronal respecto a la señal sin procesar, también incluida a modo de comparativa.

Cabe recalcar que, al igual que para el apartado 4.3, en este no se han encontrado irregularidades con las señales de 10 dB de SNR. Esto demuestra que la anomalía procede de la utilización en conjunto de las bases de datos usadas habitualmente, TIMIT y HU.

Con lo visto anteriormente, se puede suponer que o bien el idioma castellano o bien la base de datos ALBAYZIN resultan más conveniente para tareas de limpieza de ruido que el inglés o la propia base de datos TIMIT.

	SNR				
	-5	0	5	10	15
Media de MOS Audios Sucios Albayzin	1,2835	1,5919	1,9013	2,2365	2,5861
Media de MOS Audios Limpios Albayzin 50 epochs / 90h	2,2124	2,5324	2,7858	2,9942	3,173

Tabla 5: Resultados de media de PESQ para audios ruidosos de Albayzin y audios limpiados con 90h y 50 epochs.

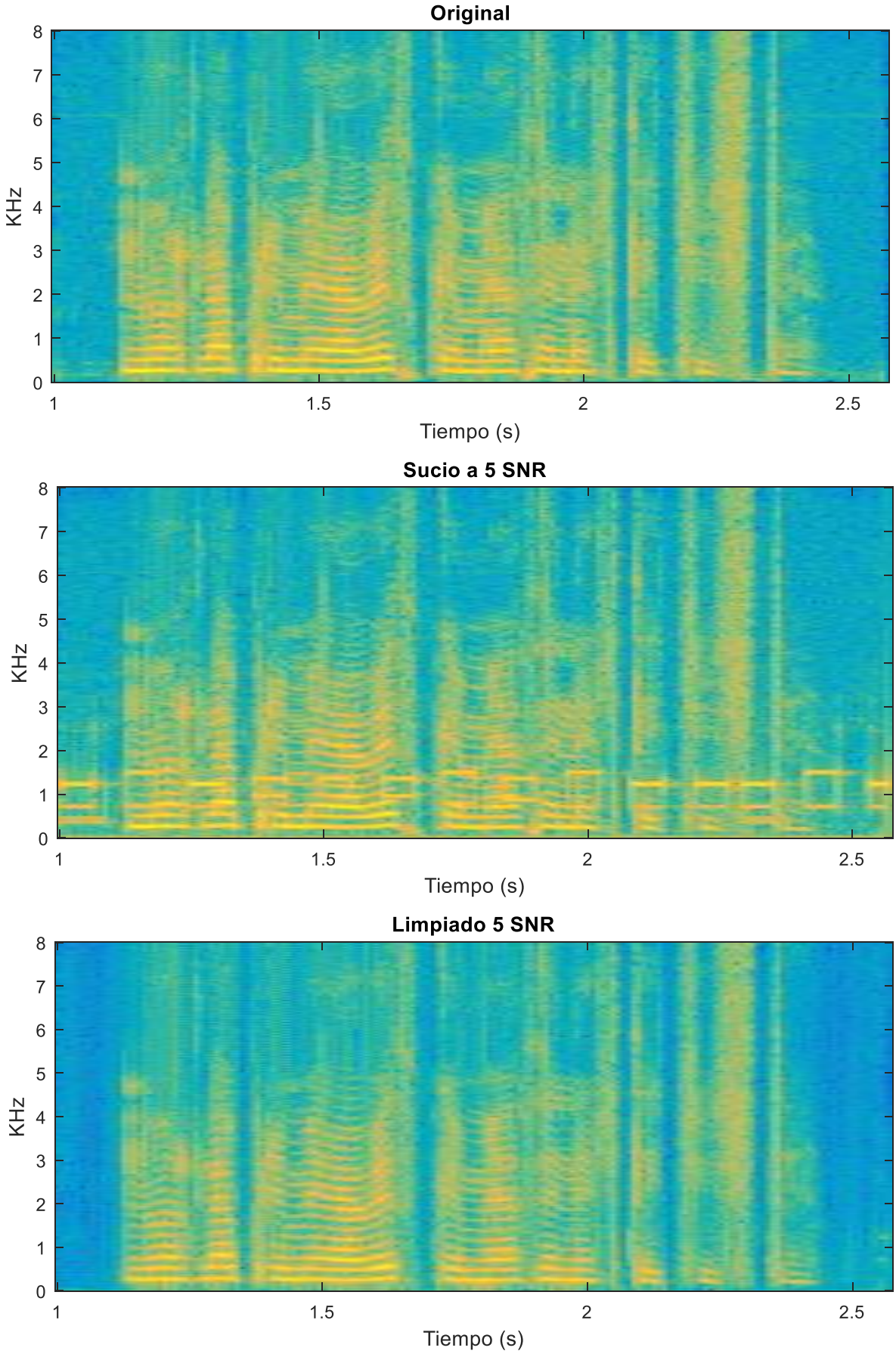

Figura 15: Ejemplo de Espectrogramas con base de datos de voz ALBAYZIN: original (arriba), ruidoso original (medio) y procesado por la red neuronal (abajo).

5. Conclusiones y trabajo futuro

A modo de conclusión general se puede decir que con este proyecto se ha conseguido desarrollar un sistema basado en una red neuronal capaz de eliminar de una manera satisfactoria buena parte del ruido presente en señales de voz. De una manera más parcial se pueden destacar diferentes logros:

- Se ha implementado con éxito un sistema de enventanado, generación de entorno y reconstrucción temporal de señales de voz.

- Se ha conseguido generar una red neuronal funcional para entrenamientos con bases de datos de tamaño medio, hasta 10h, mediante el uso de Theano y Blocks. El código generado para este problema puede ser reutilizado en la implementación de redes neuronales de cualquier tamaño y finalidad puesto que ha sido escrito de la manera más general posible.

- Se desarrolló un sistema de entrenamiento de redes neuronales funcional con bases de datos de gran tamaño, superando así las limitaciones de recursos de memoria en el pc utilizado. Este código también presenta un carácter generalista y por lo tanto puede reutilizarse en problemas de otra finalidad.

- Se llevaron a cabo pruebas de limpieza, con señales ruidosas a varios niveles de SNR, consiguiendo en estas resultados objetivos mejores que los aportados por el sistema PESQ con las señales ruidosas originales.

  Nuestros resultados no son tan buenos como los que se pueden encontrar en el artículo de referencia '*A Regression Approach to Speech Enhancement Based on Deep Neural Netwroks*'. Esto se puede deber a que en el artículo se incluían sistemas de ajuste adicionales como una normalización y desnormalización de media y varianza y una técnica de ajuste de la varianza de salida que no se han incluido en este trabajo.

- Se han realizado tests con bases de datos de ruidos nunca antes vistos; por las redes entrenadas, con resultados muy satisfactorios. A su vez se realizaron limpiezas de audios pertenecientes a una base de datos de voz de un idioma distinto al de entrenamiento con, también, grandes resultados.

  Algo a recalcar es que esto es algo propio y novedoso de este TFG puesto que con ello se ha ampliado considerablemente la variedad de pruebas con diferentes escenarios, no tratados anteriormente, respecto a los propuestos en el artículo de referencia.

Por otro lado, y para finalizar, se pueden proponer los siguientes puntos como trabajo a futuro:

- Extender el entrenamiento a un número mayor de horas.

- Preprocesado con normalización y desnormalización de bases de datos para entrenamiento de una red neuronal para el mismo objetivo.

- Pruebas de limpieza con idiomas distintos a los de entrenamiento, más allá de inglés y castellano.

- Entrenamiento y limpiezas con bases de datos multi-idioma.

6. Referencias

[1] LeCun, Y., Bengio, Y. & Hinton, G. (eds.), "Deep Learing", Nature, Volumen 521, pp 436-444, 28 Mayo 2015.

[2] Xu, Y., Du, J., Dai, L, & Lee, C. (eds.), "A Regression Approach to Speech Enhancement Based on Deep Neural Networks", IEEE Transactions on Audio, Speech, ond Language Processing, Volumen 23, nº1, pp 7-19, Enero 2015.

[3] González Caravaca, G, "Reducción de Ruido en Grabaciones de Voz", Proyecto Fin de Carrera, EPS-UAM, Junio 2011.

[4] Chen, J., Benesty, J., Huang E., Y., Diethorn, J., Fundamentals of Noise Reduction, in "Springer Handbook of Speech Processing", pp 866-868, 2008.

[5] Maroñas Molano, J., "Adversarial Learning with Ladder Networks", Proyecto Fin de Máster, Universitat Politècnica de Valencià, 18 Septiembre 2016.

[6] "Neural Networks", material académico, Universitat Politècnica de Valencià.

[7] Bishop, C. M., "Neural Networks for Pattern Recognition", pp. 195-200, 1995

[8] Documentación TIMIT. (1990). The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus.

7. Glosario

DNN: Deep Neural Network

SNR: Signal to Noise Ratio

MMSE: Minimun Mean Square Error

HMM: Hidden Markov Model

DFT: Discrete Fourier Transform

TFG: Trabajo Fin de Grado

ITU: International Telecommunication Unit

ReLU: Rectified Linear Unit